

THE MOSER-TARDOS FRAMEWORK WITH PARTIAL RESAMPLING

DAVID G. HARRIS¹ AND ARAVIND SRINIVASAN²

ABSTRACT. The resampling algorithm of Moser & Tardos is a powerful approach to develop constructive versions of the Lovász Local Lemma (LLL). We generalize this to a *partial* resampling approach: when a bad event holds, we resample an appropriately-random *subset* of the variables that define this event, rather than the entire set as in Moser & Tardos. This is particularly useful when the bad events are determined by sums of random variables. This leads to several improved algorithmic applications in scheduling, graph transversals, packet routing etc. For instance, we improve the approximation ratio of a generalized D -dimensional scheduling problem studied by Azar & Epstein from $O(D)$ to $O(\log D / \log \log D)$, and settle a conjecture of Szabó & Tardos on graph transversals asymptotically. As a point of comparison with the MT algorithm, we show a family of constraint satisfaction problems that does not have “locality” (every constraint is affected by every variable), and so the LLL and the Moser-Tardos algorithm do not give any useful results; however, our resampling approach gives strong scale-free approximation ratios and terminates in expected polynomial time for this family.

1. INTRODUCTION

The Lovász Local Lemma (LLL) [8] is a fundamental probabilistic tool. The breakthrough of Moser & Tardos shows that a very natural *resampling* approach yields a constructive approach to the LLL [23]; this, along with a few subsequent investigations [9, 18], gives a fairly comprehensive suite of techniques to develop algorithmic versions of the LLL. The basic algorithm of [23] is as follows. Suppose we have “bad” events B_1, B_2, \dots, B_m , each B_i being completely determined by a subset S_i of *independent* random variables X_1, X_2, \dots, X_ℓ . Then, assuming that the standard sufficient conditions of the LLL hold, the following resampling algorithm (which we refer to as the *MT algorithm*) quickly converges to a setting of the X_j ’s that simultaneously avoids all the B_i :

- first sample all the X_j ’s (independently) from their respective distributions;
- **while** some bad event is true, pick one of these, say B_i , arbitrarily, and resample (independently) all the variables X_j for $j \in S_i$.

We generalize this to a *partial resampling* approach, which we simply call the Partial Resampling Algorithm (PRA); the idea is to carefully choose a distribution D_i over *subsets* of S_i for each i , and then, every time we need to resample, to first draw a subset from D_i , and then only resample the X_j ’s that are contained in this subset. This partial-resampling approach leads to algorithmic results for many applications that are not captured by the LLL.

Conference versions of this work. Preliminary versions of parts of this paper appeared in two papers by the authors: [10, 11].

In order to motivate our applications, we start with two classical problems: scheduling on unrelated parallel machines [21], and low-congestion routing [26]. In the former, we have n jobs and m machines (we interchange the standard use of the indices i and j here in order to conform to the rest of our notation), and each job i needs to be scheduled on any element of a given subset

¹Department of Computer Science, University of Maryland, College Park, MD 20742. Research supported in part by NSF Awards CNS-1010789 and CCF-1422569. Email: davidgharris29@gmail.com.

²Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported in part by NSF Awards CNS-1010789 and CCF-1422569. Email: srin@cs.umd.edu.

L_i of the machines. If job i is scheduled on machine j , then j incurs a given load of $p_{i,j}$. The goal is to minimize the *makespan*, the maximum total load on any machine. The standard way to approach this is to introduce an auxiliary parameter T , and ask if we can schedule with makespan T [21, 30]. Letting $[k]$ denote the set $\{1, 2, \dots, k\}$, a moment's reflection leads to the following integer-programming formulation:

$$\begin{aligned}
(1) \quad & \forall i \in [n], \sum_{j \in L_i} z_{i,j} = 1; \\
(2) \quad & \forall j \in [m], \sum_i p_{i,j} z_{i,j} \leq T; \\
(3) \quad & \forall (i, j), p_{i,j} > T \implies z_{i,j} = 0; \\
(4) \quad & \forall (i, j), z_{i,j} \in \{0, 1\}.
\end{aligned}$$

(Although (3) is redundant for the IP, it will be critical for the natural LP relaxation [21].)

In low-congestion routing, we are given a collection of (source, destination) pairs $\{(s_i, t_i) : i \in [n]\}$ in a V -vertex, K -edge directed or undirected graph G with edge-set E ; each edge $f \in E$ has a capacity c_f , and we are also given a collection $P_i = \{P_{i,j}\}$ of possible routing paths for each (s_i, t_i) -pair, with each such path indexed by i and an auxiliary index j . We aim to choose one path from P_i for each i , in order to minimize the *relative congestion*: the minimal T such that the maximum load on any edge f is at most $T \cdot c_f$. We get a similar IP formulation:

$$\text{minimize } T \text{ subject to } \left[\forall i, \sum_j z_{i,j} = 1; \forall f \in E, \sum_{(i,j): f \in P_{i,j}} z_{i,j} \leq T \cdot c_f; z_{i,j} \in \{0, 1\}. \right]$$

Our class of problems. Given the above two examples, we are ready to define the class of problems that we will study. As above, we have n variables X_1, X_2, \dots, X_n ; we need to choose one value for each variable, which is modeled by “assignment constraints” (1) on the underlying indicator variables $x_{i,j}$. In addition, we have a set \mathcal{B} of (undesirable) events; these are all conjunctions of the elementary events $X_i = j$; we aim to choose the variables X_1, \dots, X_n , and such that all the $B \in \mathcal{B}$ are falsified. It is easily seen that these two applications have the undesirable events defined by *linear* threshold functions of the form “ $\sum_{i,j} a_{k,i,j} [X_i = j] > b_k$,” where $[X_i = j]$ is the indicator that variable X_i takes on value j ; we will also consider bad-events which are non-linear, some of which will be crucial in our packet-routing application. We develop a *partial resampling* approach to our basic problem in Section 2; Theorem 2.6 presents some general conditions under which our algorithm quickly computes a feasible solution X_1, \dots, X_n .

The probabilistic analysis of MT and related algorithms is governed by *witness trees*. While these are easy to count when all bad-events are essentially the same (the “Symmetric LLL”), this can be complicated in the more general (“asymmetric”) case.

A key technical tool in our analysis is a new formula for counting the witness trees. This greatly simplifies the analysis of the Asymmetric LLL. It is critical to obtaining usable formulas for complicated applications of the Partial Resampling framework, but it is also very useful for analyzing the standard Moser-Tardos framework.

Let us next motivate our result by describing three families of applications.

1.1. The case of non-negative linear threshold functions. In the scheduling and routing applications, the bad events B_k are defined by non-negative linear threshold function: the constraints (i.e., the complements of the B_k) have the form

$$(5) \quad \sum_{i,j} a_{k,i,j} [X_i = j] \leq b_k.$$

(The matrix A of coefficients $a_{k,i,j}$ here, has K rows indexed by k , and some N columns that are indexed by pairs (i, j) .) Recall that all our problems will have the assignment constraints (1) as well. There are two broad types of approaches for such problems, both starting with the natural LP relaxation of the problem, wherein we allow variables $z_{i,j}$ to be the *fractional* assignment of variable X_i to value j . Suppose the LP relaxation has a solution $\{z_{i,j}\}$ which satisfies for all $k = 1, \dots, K$ the constraint $\sum_{i,j} a_{k,i,j} z_{i,j} \leq c_k$, where $c_k \leq b_k$ for all k ; by scaling, we will assume throughout that $a_{k,i,j} \in [0, 1]$. The natural question is:

“What conditions on the matrix A and vectors b, c ensure that there is an integer solution that satisfies (1) and (5), which, furthermore, can be found efficiently?”

The first of the two major approaches to this is polyhedral. Letting D denote the maximum column sum of A , i.e., $D = \max_{i,j} \sum_k a_{k,i,j}$, the rounding theorem of [17] shows constructively that for all k ,

$$(6) \quad b_k = c_k + D$$

suffices. The reader is asked to verify that given a solution to the LP-relaxation of makespan minimization that satisfies (1), (2) and (3), bound (6) implies that we can find a schedule with makespan at most $2T$ efficiently. This 2-approximation is the currently best-known bound for this fundamental problem, and what we have seen here is known to be an alternative to the other polyhedral proofs of [21, 30].

The second approach to our problem is randomized rounding [26]: given an LP-solution z , choose exactly one j independently for each i , with the probability of $z_{i,j}$ of setting variable i equal to j . The standard “Chernoff bound followed by a union bound over all K rows” approach [26] shows that, in order for our goal to be achieved with probability at least $1/2$, we may set

$$(7) \quad b_k = C \cdot \frac{\log K}{\log(2 \log K / c_k)} \text{ if } c_k \leq \log K; \quad b_k = c_k + C \cdot \sqrt{c_k \cdot \log K} \text{ if } c_k > \log K$$

where C is some universal constant. In particular, the low-congestion routing problem can be approximated to within $O(\frac{\log K}{\log \log K})$ in the worst case, where K denotes the number of edges.

Let us compare these known bounds (6) and (7). The former is good when all the c_k are “large” (say, much bigger than, or comparable to, D – as in the 2-approximation above for scheduling); the latter is better when D is too large, but unfortunately does not exploit the sparsity inherent in D – also note that $K \geq D$ always since the entries $a_{k,i,j}$ of A lie in $[0, 1]$. A natural question is whether we can interpolate between these two: especially consider the case (of which we will see an example shortly) where, say, all the values c_k are $\Theta(1)$. Here, (6) gives an $O(D)$ -approximation, and (7) yields an $O(\frac{\log K}{\log \log K})$ -approximation. Can we do better? We answer this in the affirmative in Theorem 6.1 – we are able to essentially replace K by D in (7), by showing constructively that if we have $c_k = R \geq 1$, then we achieve

$$b_k = C \frac{\log(D+1)}{1 + \log \frac{\log(D+1)}{R}} \text{ if } R \leq \log(D+1); \quad b_k = R + C \sqrt{R \log(D+1)} \text{ if } R \geq \log(D+1)$$

suffices.

We will show in Section 6.4 that such “scale-free” bounds (that is, b_k is a function of R, D but not of the n or K) are impossible for the MT algorithm. This is true even for systems whose constraint matrix has only $\{0, 1\}$ entries, and even when $D = 1$, and even when b_k is an arbitrarily large function of R . We will see cases in which the MT algorithm is no better than random search, requiring exponential time, while our Partial Resampling Algorithm is able to achieve (1.1) simply and efficiently.

Application to multi-dimensional scheduling. Consider the following D -dimensional generalization of scheduling to minimize makespan, studied by Azar & Epstein [4]. Their $(D+1)$ -approximation here also holds for the following generalization, and again follows quickly from (6).

Here, when job i gets assigned to machine j , there are D dimensions to the load on j (say runtime, energy, heat consumption, etc.): in dimension ℓ , this assignment leads to a load of $p_{i,j,\ell}$ on j (instead of values such as $p_{i,j}$ in [21]), where the numbers $p_{i,j,\ell}$ are given. Analogously to (1), (2) and (3), we ask here: *given a vector (T_1, T_2, \dots, T_D) , is there an assignment that has a makespan of at most T_ℓ in each dimension ℓ ?* The framework of [4] and (6) gives a $(D+1)$ -approximation¹ while our bound (1.1) yields an $O(\frac{\log D}{\log \log D})$ -approximation; since $D \ll K$ typically in this application, this is also a significant improvement over the $O(\frac{\log K}{\log \log K})$ -approximation that follows from (7).

Comparison with other known bounds. As described above, our bound (1.1) improves over the two major approaches here. However, two related results deserve mention. First, a bound similar to (1.1) is shown in [19, 10], but with D' , the maximum number of nonzeros in any column of A , playing the role of D . Note that $D' \geq D$ always, and that $D' \gg D$ is possible. Moreover, the bound of [19] primarily works when all the c_k are within an $O(1)$ factor of each other, and rapidly degrades when these values can be disparate; the bound of [10] is nonconstructive. Finally, our bound is better when $R \gg \log(D+1)$; other works give the bound in this case of $b_k = R + C_0 \sqrt{R \log R}$.

1.2. Transversals with omitted subgraphs. Given a partition of the vertices of an undirected graph $G = (V, E)$ into blocks (or *classes*), a *transversal* is a subset of the vertices, one chosen from each block. An *independent transversal*, or independent system of representatives, is a transversal that is also an independent set in G . The study of independent transversals was initiated by Bollobás, Erdős & Szemerédi [6], and has received a considerable amount of attention (see, e.g., [1, 2, 3, 14, 15, 16, 22, 31, 32]). Furthermore, such transversals serve as building blocks for other graph-theoretic parameters such as the linear arboricity and the strong chromatic number [2, 3]. We improve (algorithmically) a variety of known sufficient conditions for the existence of good transversals, in Section 5. In particular, Szabó & Tardos present a conjecture on how large the blocks should be, to guarantee the existence of transversals that avoid K_s [31]; we show that this conjecture is true asymptotically for large s . We also study weighted transversals, as considered by Aharoni, Berger & Ziv [1], and show that near-optimal (low- or high-) weight transversals exist, and can be found efficiently. In particular, we improve the quantitative bounds of [10] and show that “large-weight” (existentially-optimal) independent transversals exist, once the smallest block-size becomes reasonably large.

1.3. Packet routing with low latency. A well-known packet-routing problem is as follows. We are given an undirected graph G with N packets, in which we need to route each packet i from vertex s_i to vertex t_i along a *given simple path* P_i . The constraints are that each edge can carry only one packet at a time, and each edge traversal takes unit time for a packet; edges are allowed to queue packets. The goal is to conduct feasible routings along the paths P_i , in order to minimize the *makespan* T , the relative of the scheduling notion above that refers to the time by which all packets are delivered. Two natural lower-bounds on T are the *congestion* C (the maximum number of the P_i that contain any given edge of G) and the *dilation* D (the length of the longest P_i); thus, $(C+D)/2$ is a universal lower-bound, and there exist families of instances with $T \geq (1 + \Omega(1)) \cdot (C+D)$ [27]. A seminal result of [20] is that $T \leq O(C+D)$ for all input instances, using constant-sized queues at the edges; the asymptotic notation hides a rather large constant. Building on further improvements [28, 25], our work [10] developed a nonconstructive $7.26(C+D)$ and a constructive $8.84(C+D)$ bound; we improve these further to a constructive $6.78(C+D)$ here.

Informal discussion of the Partial Resampling Algorithm. To understand the intuition behind our Partial Resampling Algorithm, consider the situation in which constraints of the form

¹As usual in this setting, an “approximation” here is an algorithm that either proves that the answer to this question is negative (by showing that the LP is infeasible), or presents the desired approximation simultaneously for each T_ℓ .

$[X_{i_1} = j_1] + \dots + [X_{i_k} = j_k] \leq t$, where the expected value of $[X_{i_1} = j_1] + \dots + [X_{i_v} = j_v]$ is $\mu \leq t$. (We use the Iverson notation here and throughout the paper, so that $[X_i = j]$ is the indicator variable for the event that $X_i = j$.) There are many ways to analyze this situation via the LLL and MT, but let us consider the simplest situation in which we have a distinct bad-event for each constraint.

Now, suppose we wish to run the MT algorithm on this problem. We begin by drawing all the variables X_1, \dots, X_n from their original distribution, and we encounter some true bad-event $[X_{i_1} = j_1] + \dots + [X_{i_k} = j_k] > t$. At this point, the MT algorithm would resample all of the variables affected by this event — that is, all the variables X_{i_1}, \dots, X_{i_k} . This could be a large number of variables.

But, now suppose that we have $X_{i_l} \neq j_l$ for some $l = 1, \dots, k$. In that case, variable X_{i_l} seems like it is “helpful” for this bad-event. Since the goal of resampling a bad-event is to “fix” it, then resampling this X_{i_l} seems counter-productive. Thus, heuristically at least, it seems more appropriate to only resample the variables with $X_{i_l} = j_l$. In the original step of the MT algorithm, the expected number of such variables is μ , and we expect that in intermediate stages of the MT algorithm it should also be close to μ . Thus, heuristically, we should only be resampling about μ variables, not all k variables.

In fact, even this is somewhat too many variables to resample. Since we expect about μ variables to have $X_{i_l} = j_l$, it is only the $t - \mu$ “extra” variables which are causing the bad-event to occur. In this sense, a variable X_i is causing the bad-event only if it is “the straw that breaks the camel’s back,” that is, if it is the key variable which brings the sum over the threshold t . Any individual variable X_i only has a small chance of being a guilty variable, even smaller than μ .

The power of the partial resampling approach comes from the fact it makes steady progress toward a solution — when there is a bad-event, we make a minimal change to fix it, while preserving as much of the prior solution as possible. For these linear-threshold bad-events, it is only these few, guilty variables which should be resampled. We thus make much smaller steps to fix any bad-event, making steady progress toward a solution which avoids them all.²

Comparison with the MT algorithm. In Section 6.4, we will see a rather extreme version of the above phenomenon. We construct a family of constraint satisfaction problems, defined by linear threshold constraints $[X_{i_1} = j_1] + \dots + [X_{i_k} = j_k] \leq t$, in which *every* constraint is affected by *every* variable. In this case, there is no “locality” (in the sense of the LLL). The MT algorithm completely throws away its partial solution at every iteration, and starts from scratch. So it is not really doing any useful work. Not surprisingly, the MT algorithm cannot guarantee *any* scale-free approximation factors (independent of the number of constraints or variables) for this type of problem. We will show, however, that the partial-resampling approach yields a very good approximation in expected polynomial time.

A less extreme version of this is seen also in Section 5; here the LLL applies, albeit with weaker parameters than our Partial Resampling Algorithm.

In general, the Partial Resampling Algorithm tends to work well when there are common configurations, which are not actually forbidden, but are nonetheless “bad” in the sense that they increase the probability of a bad-event. In the case of a sum of random variables, for example, this occurs whenever $X_{i_l} = j_l$. We will see other examples of more complicated types of bad-but-legal configurations.

²There is an alternative way to apply the LLL in this context, which is to define a separate bad-event for each *atomic* bad configuration, that is, for each subset of variables which exceed value of t . This subdivides the original bad-event into approximately $\binom{n}{t}$ separate smaller bad-events. This approach can be effective in some regimes, especially when $t \gg \mu$, and can lead to scale-free configurations. However, this method suffers from the drawback that the sum of the probabilities of these atomic bad-events is much larger than the original probability of the single bad-event (because these atomic events has significant positive correlation). The method we develop will be strictly stronger than this approach.

Organization of the paper. The Partial Resampling Algorithm is discussed in detail in Section 2. We give a criterion, similar to the asymmetric LLL, for showing that this algorithm terminates in expected polynomial time.

Section 3 describes an alternative, more succinct formulation of the “weighting function” necessary to analyze the PRA. This method is very useful for the standard MT algorithm as well, but it is particularly important for the PRA, because the criterion we develop in Section 2 has a huge number of parameters and is difficult to work with directly.

Section 4 shows how to apply the PRA when the underlying bad-events are linear threshold functions. Such events are ubiquitous in combinatorics and algorithms, and the PRA can deal with them in any especially powerful way.

Applications are discussed in the following three sections: transversals with omitted subgraphs, improved integrality gaps for column-sparse packing problems, and packet routing in Sections 5, 6, and 7 respectively.

2. THE PARTIAL RESAMPLING ALGORITHM

2.1. Notation. We begin by discussing some basic definitions that will apply throughout.

We have n variables; each variable has a finite set of possible assignments, which we identify with a finite subset of the integers. We define the probability space Ω , in which the variables are assigned independently: for each variable i we set $X_i = j$ with probability $p_{i,j}$, where j ranges over the set of valid assignment to variable i . Henceforth we will not be explicit about the set of possible assignments, so we write simply $\sum_j p_{i,j} = 1$.

We refer to any ordered pair (i, j) where j is an assignment of variable i , as an *element*. We let \mathcal{X} denote the set of all elements. Given any vector $\vec{\lambda} = (\lambda_{i,j})$ indexed by elements $(i, j) \in \mathcal{X}$, we define, for any set $Y \subseteq \mathcal{X}$,

$$(8) \quad \lambda^Y = \prod_{(i,j) \in Y} \lambda_{i,j}.$$

Atomic events as sets of elements. We are also given a collection \mathcal{B} of bad-events. For our purposes, we may suppose that every bad-event $B \in \mathcal{B}$ is defined as a conjunction of terms of the form

$$B \equiv X_{i_1} = j_1 \wedge \cdots \wedge X_{i_l} = j_l$$

We refer to this type of conjunction of variable assignment conditions as an *atomic event*. We slightly abuse notation so this atomic event would be represented by the set $\{(i_1, j_1), \dots, (i_l, j_l)\}$. Thus, when we write $(i, j) \in B$, we mean that a necessary condition for bad-event B to be true is that $X_i = j$. We assume that atomic events do not contain two values from the category, that is, they do not contain both (i, j) and (i, j') for $j \neq j'$; any event which did so would be impossible.

In many applications of the LLL, the bad-events may be more complex. However, we can always write a complex bad event as a union of a (possible large) number of atomic bad-events. For example, if a bad-event is that $[X_{i_1} = j_1] + \cdots + [X_{i_v} = j_v] \geq t$, then this can be represented as $\binom{v}{t}$ separate atomic events.

Labeling bad-events. For technical reasons, which we will discuss more in Section 2.6, we may sometimes need to attach labels in the range $\{1, \dots, K\}$ to each bad-event. Thus, we suppose there is a labeling $\mathcal{L} : \mathcal{B} \rightarrow [K]$. We will use the notation

$$\mathcal{B}_k = \{B \in \mathcal{B} \mid \mathcal{L}(B) = k\}$$

To provide a very brief explanation of the role played by labels: if there are multiple complex bad-events, we often want to analyze each complex bad-event separately. By adding labels to the complex bad-event, we will be able to limit their interactions with each other. *In reading the following sections, the reader is advised to keep in mind the case when $K = 1$ (so that essentially the labeling map does not play a part). Almost all of the intuition and results apply in that setting,*

and many of our applications will use it as well. We refer to the case when $K = 1$ as the trivial labeling.

2.2. Fractional hitting-sets. In order to use our algorithm, we will need to specify an additional parameter. For each $k = 1, \dots, K$ we must specify a *fractional hitting-set* Q_k , which essentially tells us how to resample the variables in that bad event.

Definition 2.1. Suppose $C : 2^{\mathcal{X}} \rightarrow [0, 1]$ is a function mapping subsets of \mathcal{X} to real numbers in the range $[0, 1]$.

Let B be an atomic event, which we view as a set of elements $B = \{(i_1, j_1), \dots, (i_r, j_r)\}$. We say that C is a fractional hitting set for B if we have that

$$(9) \quad \sum_{Y \subseteq B} C(Y) \geq 1$$

Suppose \mathcal{B} is a set of atomic bad-events. We say that C is a fractional hitting set for \mathcal{B} if C is a fractional hitting set for all $B \in \mathcal{B}$.

If there are K possible labels, we will select K fractional hitting sets Q_1, \dots, Q_K ; for each $k = 1, \dots, K$ we require that Q_k is a fractional hitting set for \mathcal{B}_k .

Definition 2.2 (A supported event). Given any $Y \subseteq X$ and $k \in [K]$, we say that (Y, k) is supported if $Q_k(Y) > 0$.

Remark regarding Definition 2.1. We may assume, without loss of generality, that if Y contains more than one value for a variable (that is, if it contains (i, j) and (i, j')) then $Q_k(Y) = 0$.

One possible choice for the fractional hitting set is the bad-event itself, which is easily verified to be a valid fractional hitting-set:

Definition 2.3. Let \mathcal{B} be a set of bad-events. We define the trivial hitting-set for \mathcal{B} by

$$C(Y) = \begin{cases} 1 & \text{if } Y \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

If the trivial hitting-set is used, then our analysis essentially reduces to the ordinary MT algorithm. We will discuss later how to construct more powerful types of fractional hitting-sets.

2.3. Partial Resampling Algorithm and the Main Theorem. Consider the following relative of the MT algorithm, which we refer to as the *Partial Resampling Algorithm (PRA)*:

1. Select one value for each variable $i = 1, \dots, n$. The probability of selecting $X_i = j$ is $p_{i,j}$.
2. Repeat the following, as long as there is some true bad-event $B \in \mathcal{B}$:
 3. Select, arbitrarily, some true bad-event $B \in \mathcal{B}$. We refer to this set B as the *violated set*.
 4. Select exactly one subset $Y \subseteq B$. The probability of selecting a given Y is given by

$$P(\text{select } Y) = \frac{Q_k(Y)}{\sum_{Y' \subseteq B} Q_k(Y')}$$

where $k = \mathcal{L}(B)$. We refer to Y as the *resampled set*.

5. Resample all the variables involved in Y independently, using the vector p .

The main difference between the PRA and the MT algorithms is that in MT, if there is a bad event that is currently true, we would resample *all* the variables which it depends upon. Here, we only resample a (carefully-chosen, random) subset of these variables.

We will need to keep track of the dependency graph corresponding to our fractional hitting set. This is more complicated than the usual Moser-Tardos setting, because we will need to distinguish

two ways that subsets of elements Y, Y' could affect each other: they could share a variable, or they could both be potential resampling targets for some bad-event. In the usual LLL analysis, we only need to keep track of the first type of dependency. The following symmetric relation \approx (and its two supporting relations \sim and \bowtie) will account for this:

Definition 2.4. (Symmetric relations \sim, \bowtie_k , and \approx) Let $Y, Y' \subseteq \mathcal{X}$.

We say $Y \sim Y'$ iff there exists a triple (i, j, j') such that $(i, j) \in Y$ and $(i, j') \in Y'$: i.e., iff Y and Y' overlap in a variable. We also write $i \sim Y$ (or $Y \sim i$) to mean that Y involves variable i (i.e., $(i, j) \in Y$ for some j .)

For each integer k , we say $Y \bowtie_k Y'$ iff $Y \not\sim Y'$ and there is some event $B \in \mathcal{B}_k$ with $Y, Y' \subseteq B$.

Relation \approx is defined between pairs (Y, k) : We define $(Y, k) \approx (Y', k')$ iff $Y \sim Y'$ or $Y \bowtie_k Y'$. Note that, by definition, it is impossible for both to occur simultaneously.

We say that the \bowtie relations are null if for all Y, Y' with $Q_k(Y) > 0, Q_k(Y') > 0$ we have $Y \not\bowtie_k Y'$.

We give three conditions for the PRA to terminate. These conditions are analogous to, respectively, the cluster-expansion LLL criterion [5], the asymmetric LLL, and the symmetric LLL. In order to state the conditions, we introduce some definitions:

Definition 2.5. (A neighbor-set for (Y, k)) Given any $Y \subseteq \mathcal{X}$ and $k \in [K]$ and a set $\mathcal{T} \subseteq 2^{\mathcal{X}} \times [K]$, we say that \mathcal{T} is a neighbor-set for (Y, k) if the following conditions hold:

- (1) For all $(Z, l) \in \mathcal{T}$ we have $Z \approx Y$.
- (2) There do not exist $(Z, l), (Z', l') \in \mathcal{T}$ with $Z \sim Z'$
- (3) There is at most one pair of the form $(Z, k) \in \mathcal{T}$ satisfying $Z \bowtie_k Y$

Theorem 2.6. (Main Theorem) Suppose we are given fractional hitting sets Q_1, \dots, Q_K for $\mathcal{B}_1, \dots, \mathcal{B}_K$ respectively.

In each of the following three cases, the PRA terminates in a feasible configuration avoiding all bad events with probability one.

(a) Suppose there exists $\mu : 2^{\mathcal{X}} \times [K] \rightarrow [0, \infty)$ which satisfies, for all $Y \subseteq \mathcal{X}$ and all $k \in [K]$,

$$\mu(Y, k) \geq p^Y Q_k(Y) \sum_{\substack{\mathcal{T} \text{ a neighbor-set} \\ \text{for } Y}} \prod_{(Y', k') \in \mathcal{T}} \mu(Y', k')$$

Then, the expected number of resamplings is at most $\sum_{(Y, k)} \mu(Y, k)$.

(b) Suppose there exists $\mu : 2^{\mathcal{X}} \times [K] \rightarrow [0, \infty)$ which satisfies, for all $Y \subseteq \mathcal{X}$ and all $k \in [K]$,

$$\mu(Y, k) \geq p^Y Q_k(Y) \left(\prod_{Y' \sim Y} \left(1 + \sum_{k' \in [K]} \mu(Y', k') \right) \right) \left(1 + \sum_{Y'' \bowtie_k Y} \mu(Y'', k) \right)$$

Then, the expected number of resamplings is at most $\sum_{(Y, k)} \mu(Y, k)$.

(c) Suppose that for all $Y \in 2^{\mathcal{X}}, k \in [K]$ we have $p^Y Q_k(Y) \leq P$; and suppose that for all supported (Y, k) , there are at most D supported (Y', k') with $(Y', k') \approx (Y, k)$ (note that we allow here $(Y', k') = (Y, k)$.) And suppose finally that we satisfy the criterion

$$ePD \leq 1$$

Then, the expected number of resamplings is at most $e \sum_{(Y, k)} p^Y Q_k(Y)$.

Remark on notation. When we define a function $\mu : 2^{\mathcal{X}} \times [K] \rightarrow [0, \infty)$, it is convenient to define

$$\mu(Y) = \sum_{k \in [K]} \mu(Y, k)$$

for any $Y \subseteq \mathcal{X}$. Thus, we can state the criterion of Theorem 2.6(b) more concisely as:

$$\mu(Y, k) \geq p^Y Q_k(Y) \left(\prod_{Y' \sim Y} 1 + \mu(Y') \right) \left(1 + \sum_{Y'' \boxtimes_k Y} \mu(Y'', k) \right)$$

2.4. Review of the analysis of Moser & Tardos. Our analysis of our PRA mirrors the original beautiful analysis by Moser & Tardos in [23]. We begin by briefly reviewing their approach; we recommend that the reader consult [23] for a much more thorough exposition.

Suppose that we run the MT algorithm up to time T . Suppose that at time t we resample bad-event B_t . We define the *execution log* to be the complete listing of all B_1, B_2, \dots, B_T . These are not necessarily distinct.

One key idea of Moser & Tardos is to build a type of “explanation” or “history” of all variables which contribute to each resampling. For any time t , one may produce a random variable which we refer to as the *witness tree for time t* , denoted $\hat{\tau}^t$. These are all rooted labeled trees. They are produced as follows: we begin by defining $\hat{\tau}_t^t$ to consist of a single root node labeled by B_t . For $j = t-1, t-2, \dots, 1$ we then update $\hat{\tau}_j^t$ as follows. If $\hat{\tau}_{j+1}^t$ does not contain any nodes labeled by $B' \sim B_j$, then we set $\hat{\tau}_j^t = \hat{\tau}_{j+1}^t$. Otherwise, we select v to be a node of $\hat{\tau}_{j+1}^t$ which is labeled by $B' \sim B_j$, and *and has maximal depth among all such nodes*. If there are multiple nodes at the same depth, we may break ties arbitrarily. We then define $\hat{\tau}_j^t$ by adding a new node which is a child of v and is labeled by B_j . We finally define $\hat{\tau}^t = \hat{\tau}_1^t$.

In analyzing these witness trees, it is useful to distinguish between the random variable $\hat{\tau}^t$ and possible values for that variable. We say that τ is a *tree-structure* which is a rooted labeled tree which is a possible value for $\hat{\tau}^t$; we may talk, for instance, of the probabilistic event that $\hat{\tau}^t = \tau$. We adopt the convention that if τ is a rooted labeled tree and it is *impossible* for $\hat{\tau}^t = \tau$ in any execution of the MT algorithm, that we do not call τ a tree-structure. Thus, for instance, [24] showed that all the witness trees produced by the MT algorithm have the property that the labels of the depth- k nodes are independent under \sim ; thus, we assume that any tree-structure also has this property.

Given a tree-structure τ , we say that τ *appears* if $\hat{\tau}^t = \tau$ for any value $t = 1, \dots, T$. We also define the *weight* of a tree-structure: If τ has nodes labeled B_1, B_2, \dots, B_j , then we define the weight of τ by

$$w(\tau) = \prod_{j=1}^j P(B_j)$$

We now observe that if $t \neq t'$ then necessarily $\hat{\tau}^t \neq \hat{\tau}^{t'}$. This implies that

$$T \leq \sum_{\text{tree-structures } \tau} [\tau \text{ appears}]$$

(using Iverson notation). So

$$\mathbf{E}[\text{running time of MT}] \leq \sum_{\text{tree-structures } \tau} P(\tau \text{ appears})$$

The key result by Moser & Tardos is the following Witness Tree Lemma:

Lemma 2.7 (Witness Tree Lemma for MT [23]). *Suppose that τ is a tree-structure. The probability that τ appears is at $w(\tau)$*

This immediately implies that the expected running time of MT is bounded by $\sum_{\text{tree-structures } \tau} w(\tau)$. Using the LLL criterion, in any of its forms, one can then show that $\sum_{\text{tree-structures } \tau} w(\tau)$ is bounded by polynomials in the input size; this implies that the MT algorithm terminates in polynomial time.

The proof of the Witness Tree Lemma is based on a coupling construction. We imagine that before any resamplings have been performed, we construct a resampling table — that is, for each variable $i \in [n]$, we draw an infinite sequence of values $X_{i,1}, X_{i,2}, \dots$, each of which is drawn independently according to the distribution p_i . This resampling table is a static object, not a dynamic process like the MT algorithm; this makes it easier to analyze.

Now, suppose that we have a tree-structure τ . In order to τ to appear, certain entries of the resampling table are forced to specific values. For any $i \in [n]$, it must be the case that all nodes of τ labeled by B involving variable i occur as distinct levels (as the levels of any witness tree are independent). Suppose that we list them in order of greatest to least depth as B_1, B_2, \dots, B_k . Then one can observe that the first entry $X_{i,1}$ must be compatible with B_1 ; the second entry $X_{i,2}$ must be compatible with B_2 , and so on. As the entries of the resampling table are all independent, one can see that (taking a product of all $i \in [n]$) the total probability that this is satisfied is at most $w(\tau)$.

2.5. Witness Trees for the PRA. Our proof will also be based on constructing witness trees and proving that a Witness Tree Lemma holds for them. However, there will be key differences and the proofs will be more complex.

Suppose we run the PRA and encounter bad-events B_1, \dots, B_T ; for each of these we resample the set Y_1, \dots, Y_T . We define the *execution log* of this algorithm to be the listing $(Y_1, \mathcal{L}(B_1)), \dots, (Y_T, \mathcal{L}(B_T))$. It is crucial to note that we do *not* list the violated sets B themselves; we list only the resampled set and the labels of the violated set. By avoiding the explicit listing of the violated sets, we dramatically prune the space of possible witness trees; this is one of the critical ideas of our paper.

Given the execution log, we will define a type of witness tree which is akin to MT. The nodes of the witness tree will be labeled by pairs of the form (Y, k) ; we add a node (Y, k) as a child of (Y', k') if $(Y, k) \approx (Y', k')$. However there is a key difference: each node labeled (Y, k) is only allowed to have a *single* child due to the relation \bowtie_k , and has *no children* due to the relation $\bowtie_{k'}$ for $k' \neq k$.

So suppose we are given an execution log $(Y_1, k_1), \dots, (Y_T, k_T)$; for any $t \leq T$, we define the *witness tree for time t* which we denote $\hat{\tau}^t$. We begin by defining $\hat{\tau}_t^t$ which has a single root node labeled (Y_t, k_t) . For $j = t-1, t-2, \dots, 1$ we let M_j denote the nodes v of $\hat{\tau}_{j+1}^t$ with either of the two properties:

- (1) v is labeled by (Y', k') where $Y' \sim Y_j$; OR
- (2) v is labeled by (Y', k) and $Y \bowtie_k Y'$ and v does not have any child u labeled by (Y'', k) with $Y'' \bowtie_k Y'$.

If M_j is empty, then we update $\hat{\tau}_j^t = \hat{\tau}_{j+1}^t$. Otherwise, we select some node $v \in M_j$ of greatest depth in $\hat{\tau}_{j+1}^t$ (breaking ties arbitrarily), and form $\hat{\tau}_j^t$ by adding a new node which is a child of v and is labeled by (Y_j, k_j) .

Finally, we define $\hat{\tau}^t = \hat{\tau}_1^t$. By convention, we define $\hat{\tau}_s^t = \emptyset$ (the null tree, which does not contain any nodes not even the root node), when $s > t$.

Because this will come up repeatedly in our discussion, given a node v labeled by (Y, k) , we refer to a child w labeled (Y', k) with $Y \bowtie_k Y'$ as a \bowtie -child. If a node v has a \bowtie -child u we say it *is saturated by u* otherwise we say *v is unsaturated*. Evidently in this construction each node may have one or zero \bowtie -children. Thus, another way to state the second rule for $v \in M_j$ is that v is labeled by (Y', k) and $Y \bowtie_k Y'$ and v is unsaturated.

In parallel to the random variables $\hat{\tau}^t$, we define a *tree-structure* to be a rooted labeled tree which is a possible value for $\hat{\tau}^t$. We likewise define the *weight* of a tree-structure as follows:

Definition 2.8 (Weight of a tree-structure τ). Suppose that τ is a tree-structure whose nodes are labeled $(Y_1, k_1), \dots, (Y_j, k_j)$. Then the weight of τ is

$$w(\tau) = \prod_{i=1}^j p^{Y_i} Q_{k_i}(Y_i).$$

Remark. Recall the notation (8) in parsing the value “ p^{Y_i} ” above.

We will show our main Lemma 2.10, which parallels the Witness Tree Lemma of [23], connecting the witness trees to the execution logs. However, its proof is much more involved than in [23], since we can only partially conduct the coupling with the resampling table. Instead we must analyze the evolution of the witness tree in a truly dynamic fashion. To this end, we introduce a key result which allow us to deduce some information about the relative temporal orderings of the resamplings of $\hat{\tau}^t$. Before the formal proof, we give some intuition first.

An intuitive explanation for the role of \bowtie . Suppose that we have a tree-structure τ which consists of a singleton node (Y, k) . To keep the discussion simple suppose that $K = 1$, and so the labeling plays no role; we omit k for the remainder of this intuitive explanation.

Our goal is to show that τ appears with probability at most $w(\tau) = p^Y Q(Y)$. It is not hard to see that the initial sampling of the variables must make Y true, and this event has probability p^Y (this is the same as the proof in Moser & Tardos). We would next like to say that the probability that Y was selected to be resampled is at most $Q(Y)$, giving us our probabilistic bound.

Now, on *any given instance in which Y is eligible*, the probability that we select Y is indeed at most $Q(Y)$. However, there may have been a long sequence of bad-events in which Y was eligible, yet it was not selected. If there are enough of these opportunities, then the probability that Y is eventually selected approaches to one. Thus in order to obtain a useful bound on the probability of selecting Y one must distinguish in advance a *specific* instance in which Y is eligible.

Now observe that if Y is eligible to be selected for some atomic bad-event B , but instead some other Y' is selected, then we will have $Y' \approx Y$; it may be that $Y' \sim Y$, and if not we have $Y' \bowtie_k Y$ as both $Y, Y' \subseteq B$. So if Y is eligible to be selected, but some $Y' \neq Y$ is selected instead, then Y' would be added as a child of Y in the witness tree. As τ is a singleton node labeled Y , then *a necessary condition for τ to appear is that the first time in which Y is eligible to be selected, it is indeed selected*. With some thought, one can see that this event has probability at most $Q(Y)$.

The reader should bear this intuition in mind for the remainder of the proof. Proposition 2.9 extends this to larger tree-structures, which can have more complex interactions.

Proposition 2.9. Suppose that at time s the violated set for the PRA is B with $\mathcal{L}(B) = k$. Define J to be the set of leaf nodes of $\hat{\tau}_s^t$ whose label is of the form (Y, k) for some $Y \subseteq B$. Then:

- (1) J cannot contain two nodes which are both at greatest depth (that is, $v, v' \in J$ and $\text{depth}(v) \geq \text{depth}(u)$ for all $u \in J$);
- (2) If $J \neq \emptyset$ and v is the (unique) vertex in J of greatest depth, then v must have label (Y, k) where Y is the resampled set at time s and $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t - v$
- (3) If $J = \emptyset$, then $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t$.

Proof. To show (1), observe that if there are two leaf nodes v, v' labeled (Y, k) and (Y', k) and $(Y, k) \approx (Y', k')$, then the earlier occurring node would be eligible to be placed as a child of the later occurring node. (These nodes are unsaturated as they are leaf nodes.) So the earlier would be placed at greater depth than the later one. Also, observe that if $Y \subseteq B$ and $Y' \subseteq B$ then this implies that $Y \bowtie_k Y'$ so $(Y, k) \approx (Y', k')$.

To show (2), suppose that J contains a leaf node v labeled (Y, k) , but $Y' \neq Y$ is the resampled set at time s . We must have $v \in \hat{\tau}_{s+1}^t$ (the only node in $\hat{\tau}_s^t - \hat{\tau}_{s+1}^t$ could have label (Y', k)), and v is unsaturated in $\hat{\tau}_{s+1}^t$ (it is a leaf node). As both $Y, Y' \subseteq B$, we have that $(Y', k) \approx (Y, k)$, and so

$v \in M_s$. So $\hat{\tau}_s^t$ would have a new node labeled (Y', k') at greater depth than v . This contradicts that $v \in J$.

Thus, we have shown that Y is the resampled set at time s . Observe that $\hat{\tau}_s^t$ is either equal to $\hat{\tau}_{s+1}^t$, or has a single leaf node added to $\hat{\tau}_{s+1}^t$ labeled (Y, k) . Suppose that $\hat{\tau}_{s+1}^t$ contains r leaf nodes labeled (Y, k) . All such nodes would be in M_s . This implies that a new node in $\hat{\tau}_s^t$ would be placed at greater depth than any other leaf node labeled (Y, k) . As J contains all the leaf nodes labeled (Y, k) , it must be that the only possibility for the new node added at time s is v itself. So either $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t - v$ or $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t$. If $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t - v$ we are done.

So suppose $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t$. So $\hat{\tau}_{s+1}^t$ contains $r \geq 1$ nodes labeled (Y, k) . These nodes are all in M_s so $M_s \neq \emptyset$. This implies that $\hat{\tau}_s^t$ has a new node added, which contradicts that $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t$.

To show (3), suppose that $\hat{\tau}_s^t \neq \hat{\tau}_{s+1}^t$. Then $\hat{\tau}_s^t$ has a new leaf node v added labeled (Y, k) where $Y \subseteq B$. This node v would be in J , contradicting that $J = \emptyset$. \square

We are now ready to prove the Witness Tree Lemma

Lemma 2.10 (Witness Tree Lemma). *Let τ be any tree-structure. The probability that τ appears is at most $w(\tau)$.*

Proof. We claim the following stronger result: suppose that T is any positive integer, and suppose we condition on the full resampling table as well as the full state of the stochastic process of the PRA at some time $s \leq T$. Then we have that

$$(10) \quad P\left(\bigvee_{s \leq t < T} \hat{\tau}_s^t = \tau\right) \leq \prod_{\substack{\text{nodes } v \text{ in } \tau \\ \text{labeled } (Y, k)}} Q_k(Y)$$

To simplify the notation, for any node v of τ labeled by (Y, k) , we define $Q(v) = Q_k(Y)$. So a simplified way to write (10) is $P(\bigvee_{s \leq t < T} \hat{\tau}_s^t = \tau) \leq \prod_{v \in \tau} Q(v)$.

We prove this by induction backward on s . The base case is $s = T$; here there are two cases. If $\tau = \emptyset$, then the RHS of (10) is equal to one so this is vacuously true. If $\tau \neq \emptyset$, then it is impossible to have $\hat{\tau}_s^t = \tau$ and so the LHS of (10) is equal to zero, and again the bound holds.

We now move on to the induction step. Now, suppose that we condition on the state of the PRA stochastic process at time s as well as the full resampling table, and that at time s the violated set is B with $\mathcal{L}(B) = k$. Define J to be the set of leaf nodes of τ which are labeled by (Y, k) with $Y \subseteq B$.

If J contains more than one node at greatest depth of τ , then by Proposition 2.9 it is impossible to have $\hat{\tau}_s^t = \tau$ for any $t \geq s$. So (10) holds vacuously (the LHS is equal to zero).

If J contains zero nodes at greatest depth of τ , then by Proposition 2.9 we have $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t$. So a necessary condition to have $\tau_s^t = \tau$ is to have $\tau_{s+1}^t = \tau$ for some t in the range $s + 1 \leq t < T$. By induction hypothesis, this has probability $\leq \prod_{v \in \tau} Q(v)$ as desired.

So finally suppose that J contains a single greatest-depth node v labeled (Y, k) . Then by Proposition 2.9 we must select Y for the resampled set at time s and we must have $\hat{\tau}_{s+1}^t = \hat{\tau}_s^t - v$. The probability of selecting Y is at most $\frac{Q_k(Y)}{\sum_{Y' \subseteq B} Q_k(Y')} \leq Q_k(Y) = Q(v)$.

In addition, we must have that $\hat{\tau}_{s+1}^t = \tau - v$. By induction hypothesis, this event has probability at most $\prod_{v' \in \tau - v} Q(v')$. Because our induction hypothesis states that this probability bound holds *conditional on the resampling table as well as all prior state*, we can multiply these two probabilities. Thus, the overall probability of both selecting Y and having $\hat{\tau}_{s+1}^t = \tau - v$, is at most

$$Q(v) \times \prod_{v' \in \tau - v} Q(v') = \prod_{v' \in \tau} Q(v')$$

and the induction again holds.

We now move on to prove the full result. One can verify that that the witness trees we have defined specify the same types of conditions on the resampling table as in the original Moser &

Tardos proof. The probability that the resampling table satisfies all the conditions implied by τ is at most $\prod_{\substack{v \in \tau \\ \text{labeled } (Y, k)}} p^Y$. Conditional on the resampling table, the probability of having $\hat{\tau}^t = \tau$ for $t \leq T$, is at most $\prod_{v \in \tau} Q(v)$. Thus, the overall probability of having $\hat{\tau}^t = \tau$ is at most

$$\prod_{\substack{v \in \tau \\ \text{labeled } (Y, k)}} p^Y Q_k(Y) = w(\tau)$$

We now take the limit as $T \rightarrow \infty$ to obtain the full result:

$$P\left(\bigvee_{t \geq 1} \hat{\tau}^t = \tau\right) = \lim_{T \rightarrow \infty} P\left(\bigvee_{t=1}^T \hat{\tau}^t = \tau\right) \leq \lim_{T \rightarrow \infty} w(\tau) = w(\tau)$$

proving our result. \square

In order to show that the PRA terminates, we must bound the total weight of tree-structure. This is very similar to the calculation in Moser & Tardos, so we give a sketch here:

Proposition 2.11. *Suppose we are given a weighting function satisfying Theorem 2.6(a). Then the total weight of all tree-structures with root labeled (Y, k) is at most $\mu(Y, k)$.*

Proof. For any $Y \subseteq X, k \in [K]$ let $T_h(Y, k)$ be the total weight of all tree-structures with root node labeled (Y, k) of height at most h . One can show by induction on h that $T_h(Y, k) \leq \mu(Y, k)$. To show this, note that the children of the root node must receive distinct labels $(Z_1, l_1), \dots, (Z_r, l_r)$, and that $\{(Z_1, l_1), \dots, (Z_r, l_r)\}$ must be a neighbor-set for (Y, k) . \square

Next, note that for any distinct times $t < t'$, the witness trees corresponding to t, t' are distinct. Thus, the expected number of resamplings is at most the expected number of tree-structures which appear, which is at most the sum of the weights of all witness trees, which is at most $\sum_{(Y, k)} \mu(Y, k)$. This immediately shows Theorem 2.6(a).

We can derive Theorem 2.6(b), by using the following method to enumerate neighbor-sets \mathcal{T} . First, we put into \mathcal{T} either one element (Y'', k) with $Y'' \bowtie_k Y$, or no such elements; this contributes a factor $\left(1 + \sum_{Y'' \bowtie_k Y} \mu(Y'', k)\right)$. Next, for any $Y' \sim Y$, we place (Y', k') into \mathcal{T} for at most one choice of k' . (Observe that \mathcal{T} cannot contain both (Y', k'_1) and (Y', k'_2) by definition of a neighbor-set). For any $Y' \sim Y$, this contributes the term $1 + \sum_{k' \in [K]} \mu(Y', k')$. This produces every neighbor-set \mathcal{T} , but some of the sets produced in this way are not neighbor-sets; thus this is an over-estimate. So we have

$$\sum_{\mathcal{T} \text{ a neighbor-set for } Y} \prod_{(Y, k') \in \mathcal{T}} \mu(Y', k') \leq \left(\prod_{Y' \sim Y} \left(1 + \sum_{k' \in [K]} \mu(Y', k')\right) \right) \left(1 + \sum_{Y'' \bowtie_k Y} \mu(Y'', k)\right)$$

Finally, we can derive Theorem 2.6(c) by setting $\mu(Y, k) = ep^Y Q_k(Y)$ for all Y, k .

2.6. Complex bad-events and the role of labels. Now that we have developed our PRA framework, we can clarify the role played by the labeling function $\mathcal{L} : \mathcal{B} \rightarrow [K]$.

Suppose, as is often the case in applying the LLL, that we have multiple *complex* bad-events; that is, it is not an atomic event (a conjunction of elements). For example, a bad-event might be defined in terms of a linear threshold $[X_{i_1} = j_1] + \dots + [X_{i_v} = j_v] > t$. Any complex bad-event can be expressed as a disjoint union (possibly exponentially sized) of atomic bad-events. For example, the linear threshold is definable as $\binom{v}{t}$ separate atomic bad-events. This decomposition is not necessarily unique, but we suppose we have fixed one particular choice for this decomposition.

So, we may write our family of bad events \mathcal{B} as $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \dots \cup \mathcal{B}_K$; here, each \mathcal{B}_k is a complex bad-event, which is a set of atomic bad-events.

At this point, we might want to analyze each \mathcal{B}_k separately, deriving an appropriate fractional hitting set Q_k and computing some measure of “badness” which we can aggregate over all \mathcal{B} . When

we are applying Theorem 2.6, the linkages due to \sim are relatively easy to handle in this way — it is very similar to the situation for the usual asymmetric LLL. But the linkages due to \bowtie may become hard to handle. In particular, we may have some $B \in \mathcal{B}_k$ with $Y, Y' \subseteq B$; in this case, the sets Y, Y' become dependent with each other in a complicated, non-linear way.

As is suggested by the notation, we can decouple the events \mathcal{B}_k by defining an appropriate labeling:

$$\forall B \in \mathcal{B} \quad \mathcal{L}(B) = \min_{B \in \mathcal{B}_k} k$$

We can now state a version of the Theorem 2.6(a) which involves no non-linear interactions between the separate bad-events. These results are almost identical to Theorem 2.6(a); the only difference is that in Theorem 2.6, we assume that the sets $\mathcal{B}_1, \dots, \mathcal{B}_K$ are disjoint while here they may overlap in arbitrary ways. However, in fact the overlap between bad-events $\mathcal{B}_1, \dots, \mathcal{B}_K$ can only help us.³

Theorem 2.12. *Suppose we are given complex bad-events $\mathcal{B}'_1, \dots, \mathcal{B}'_K$ (which are not necessarily disjoint), along with fractional hitting sets Q_1, \dots, Q_K for $\mathcal{B}'_1, \dots, \mathcal{B}'_K$ respectively. Suppose there exists $\mu : 2^{\mathcal{X}} \times [K] \rightarrow [0, \infty)$ which satisfies, for all $Y \subseteq \mathcal{X}$ and all $k \in [K]$,*

$$\mu(Y, k) \geq p^Y Q_k(Y) \sum_{\mathcal{T} \text{ a neighbor-set for } Y} \prod_{(Y, k') \in \mathcal{T}} \mu(Y', k')$$

Then, the expected number of resamplings is at most $\sum_{(Y, k)} \mu(Y, k)$.

Proof. It is easy to verify that $\mathcal{B}_k \subseteq \mathcal{B}'_k$ for each $k \in [K]$. Thus, μ still satisfies Theorem 2.6(a). \square

3. ENUMERATION OF WITNESS TREES BY VARIABLES

For many applications of the PRA in which the fractional hitting-sets are relatively simple, Theorem 2.6, in one of its three forms, is sufficient. However, it can be awkward to use because it requires us to analyze the very large space $2^{\mathcal{X}} \times [K]$, and check a constraint for every Y, k . We contrast this with the LLL (in its symmetric and asymmetric forms) which requires us only to check a constraint for each *bad-event*. In this section, we will reduce the number of parameters dramatically by rephrasing the LLL criterion to give a constraint in terms of each *variable*.

In this section, we suppose that are given an assignment of non-negative real numbers $\lambda_{i,j}$ to each element $(i, j) \in \mathcal{X}$. For any variable i , define $\lambda_i = \sum_j \lambda_{i,j}$. The vector λ should be thought of as an “inflated” version of the probability vector p ; roughly speaking, $\lambda_{i,j}$ is the probability that at some point in the PRA we will set $X_i = j$. We will derive a probability distribution p and a weighting function μ from this vector, and show corresponding conditions of the vector λ to ensure that the PRA terminates.

This type of accounting is useful not just for our PRA, but for the usual LLL and MT algorithm as well. When applied to the usual MT algorithm, this will give us a simplified and slightly weakened form of the cluster-expansion criterion [5]. Nevertheless, it is stronger and simpler than the standard LLL, particularly the asymmetric LLL.

³ There is one further complication, which is really a quite minor technical point. Suppose that $\mathcal{B}_{k_1}, \mathcal{B}_{k_2}$ both contain some atomic event B . To apply Theorem 2.12, we must assign the label of B to $\min(k_1, k_2)$. This means that when we encounter bad-event B , we must use the fractional hitting set $Q_{\min(k_1, k_2)}$. This can actually present some computational difficulties. For example, suppose that during the execution of the PRA, we know that $B \in \mathcal{B}_k$ is currently true. We would like to resample $Y \subseteq B$ according to the fractional hitting set Q_k ; but it may in fact be the case that $B \in \mathcal{B}_{k'}$ for some $k' < k$. Furthermore, if K is large, it may take exponential time to decide if $B \in \mathcal{B}_{k'}$ for $k' < k$.

In fact, everything works if when we encounter a bad event $B \in \mathcal{B}_1 \cup \dots \cup \mathcal{B}_K$, we select some *arbitrary* k with $B \in \mathcal{B}_k$ and resample B according to Q_k . To prove this, we must generalize the labeling function so that, for all bad-events $B \in \mathcal{B}$, we have that $L(B)$ is a *non-empty subset* of $[K]$. But this substantially complicates the notation and proofs.

Definition 3.1. (*Values $G_{i,j}, G_i, G$ that depend on a function Q and a vector λ*) Suppose we are given an assignment of non-negative real numbers $\lambda = \lambda_{i,j}$ to each element $(i, j) \in \mathcal{X}$. For $Q : 2^{\mathcal{X}} \rightarrow [0, 1]$, recalling the notation (8), we define

$$G_{i,j}(Q, \lambda) = \sum_{Y \ni (i,j)} Q(Y) \lambda^Y$$

along with “summation notations”

$$G_i(Q, \lambda) = \sum_{Y \sim i} Q(Y) \lambda^Y = \sum_j G_{i,j}(Q, \lambda) \text{ and } G(Q, \lambda) = \sum_Y Q(Y) \lambda^Y$$

Roughly speaking, if Q is a fractional hitting set for \mathcal{B} , then $G_{i,j}(Q, p)$ is the probability that variable i takes on value j , and causes some bad-event in \mathcal{B} to occur, and is selected for resampling.

In many settings, the linkages due to \bowtie are relatively insignificant compared to the linkages due to \sim . One possible reason for this is that the \bowtie linkage becomes null; this always occurs in the MT algorithm (without partial resampling). One of the main assumptions we make in this section is that the \bowtie relations are relatively negligible. Thus, instead of carefully tracking them, we use a relatively crude estimate as follows:

Definition 3.2. (*Value S_k that depends on a vector λ*) Suppose that we are given an assignment of non-negative real numbers $\lambda_{i,j}$ to each element (i, j) . We define, for each k , the parameter $S_k(\lambda)$ as

$$(11) \quad S_k(\lambda) = \max_{Y: Q_k(Y) > 0} \sum_{Y' \bowtie_k Y} Q_k(Y') \lambda^{Y'}$$

Our main theorem is now as follows:

Theorem 3.3. (*Main Theorem in terms of λ*) Suppose we are given fractional hitting sets Q_1, \dots, Q_K for $\mathcal{B}_1, \dots, \mathcal{B}_K$ respectively and a vector λ . Suppose that we set the probability distribution for the variables by $p_{ij} = \frac{\lambda_{ij}}{\lambda_i}$.

(a) Suppose that for all $k \in [K]$ we have $S_k(\lambda) < 1$, and

$$\forall i, \lambda_i \geq 1 + \sum_k \frac{G_i(Q_k, \lambda)}{1 - S_k(\lambda)}$$

Then the PRA terminates, and the expected number of resamplings is most $\sum_i \lambda_i$.

(b) Suppose that for all $k \in [K]$, $G(Q_k, \lambda) < 1$, and

$$\forall i, \lambda_i \geq 1 + \sum_k \frac{G_i(Q_k, \lambda)}{1 - G(Q_k, \lambda)}.$$

Then the PRA terminates, and the expected number of resamplings is most $\sum_i \lambda_i$.

(c) Suppose the \bowtie_k relations are null (i.e. for supported Y_1, Y_2 we have $Y_1 \not\bowtie_k Y_2$). Suppose further that

$$\forall i, \lambda_i \geq 1 + \sum_k G_i(Q_k, \lambda)$$

Then the PRA terminates, and the expected number of resamplings is at most $\sum_i \lambda_i$.

Proof. We prove only (a). To show part (b), observe that we have $S_k(\lambda) \leq G(Q_k, \lambda)$. To show part (c), observe that if \bowtie_k is null then we have $S_k(\lambda) = 0$.

We define the probability vector by $p_{ij} = \frac{\lambda_{ij}}{\lambda_i}$. We define the weighting function μ by

$$\mu(Y, k) = \frac{\lambda^Y Q_k(Y)}{1 - S_k(\lambda)}$$

We now wish to show that this satisfies Theorem 2.6(a), that is, we have

$$(12) \quad \mu(Y, k) \geq p^Y Q_k(Y) \sum_{\substack{\mathcal{T} \text{ a neighbor-set} \\ \text{for } Y}} \prod_{(Y', k') \in \mathcal{T}} \mu(Y', k')$$

Consider some $Y = \{(i_1, j_1), \dots, (i_r, j_r)\}$ and some $k \in [K]$. If $Q_k(Y) = 0$ then (12) is satisfied vacuously. Also, we may assume that i_1, \dots, i_r are distinct (as otherwise $Q_k(Y) = 0$.) We will bound now the RHS of (12).

First, \mathcal{T} may contain at most one pair (Y', k) with $Y' \bowtie_k Y$; the total contribution of such terms is at most $1 + \sum_{Y' \bowtie_k Y} \mu(Y', k)$ which we estimate as:

$$\begin{aligned} 1 + \sum_{Y' \bowtie_k Y} \mu(Y', k) &= 1 + \frac{\sum_{Y' \bowtie_k Y} \lambda^{Y'} Q_k(Y')}{1 - S_k(\lambda)} \\ &\leq 1 + \frac{S_k}{1 - S_k(\lambda)} \quad \text{as } Q_k(Y) > 0 \\ &= \frac{1}{1 - S_k(\lambda)} \end{aligned}$$

Next, for each $(i, j) \in Y$, the set \mathcal{T} may contain at most one pair (Y', k') with $(i, j') \in Y'$. This is because if there are multiple such sets $(Y', k'), (Y'', k'')$ then we would have $Y' \sim Y''$ which violates the definition of a neighbor-set. For any fixed (i, j) in Y' , this contributes at most $1 + \sum_{Y' \sim i} \sum_{k'} \mu(Y', k')$ which we may estimate as:

$$\begin{aligned} 1 + \sum_{Y' \sim i} \sum_{k'} \mu(Y', k') &= 1 + \sum_{Y' \sim i, k'} \frac{\lambda^{Y'} Q_k(Y')}{1 - S_{k'}(\lambda)} \\ &= 1 + \sum_{k'} \frac{G_i(Q_{k'}, \lambda)}{1 - S_{k'}(\lambda)} \\ &\leq \lambda_i \quad \text{by the hypothesis of Theorem 3.3(a).} \end{aligned}$$

Putting these two estimates together, we may estimate the RHS of (12) by:

$$\begin{aligned} p^Y Q_k(Y) \sum_{\mathcal{T} \text{ a neighbor-set for } Y} \prod_{(Y', k') \in \mathcal{T}} \mu(Y', k') &\leq p^Y Q_k(Y) \times \frac{1}{1 - S_k(\lambda)} \times \prod_{(i, j) \in Y} \lambda_i \\ &= \frac{Q_k(Y) \prod_{(i, j) \in Y} p_{ij} \lambda_i}{1 - S_k(\lambda)} \\ &= \frac{Q_k(Y) \prod_{(i, j) \in Y} \frac{\lambda_{ij}}{\lambda_i} \lambda_i}{1 - S_k(\lambda)} \\ &= \frac{Q_k(Y) \lambda^Y}{1 - S_k(\lambda)} \\ &= \mu(Y, k) \end{aligned}$$

Thus, Theorem 2.6(a) holds. The expected number of resamplings is

$$\begin{aligned}
\mathbf{E}[\# \text{ resamplings}] &\leq \sum_{Y,k} \mu(Y,k) \\
&\leq \sum_{i \in [n]} \sum_{Y \sim i} \sum_k \frac{\lambda^Y Q_k(Y)}{1 - S_k(\lambda)} \\
&= \sum_{i \in [n]} \sum_k \frac{G_i(Q_k, \lambda)}{1 - S_k(\lambda)} \\
&\leq \sum_{i \in [n]} (\lambda_i - 1) \leq \sum_{i \in [n]} \lambda_i
\end{aligned}$$

The bound on the total weight of the tree-structures rooted in (Y, k) , where $Y \sim i$, follows along similar lines. □

Using the formulas developed for Theorem 3.3, we can give a few more useful bounds regarding the total weight of certain classes of tree-structures.

Proposition 3.4. *Assuming that the hypotheses of Theorem 3.3 are satisfied:*

- (1) *For any $i \in [n]$, we have that the total weight of all tree-structures rooted in any (Y, k) for $Y \sim i$ is at most $\lambda_i - 1$.*
- (2) *For any supported (Y, k) , the total weight of all tree-structures rooted in (Y', k) where $Y' \bowtie_k Y$ is at most $\frac{S_k(\lambda)}{1 - S_k(\lambda)}$.*

Proof. Recall that we have that the total weight of all tree-structures rooted in any (Y, k) is at most $\mu(Y, k) = \frac{\lambda^Y Q_k(Y)}{1 - S_k(\lambda)}$. Then

To prove the first claim: we have

$$\begin{aligned}
\sum_{Y' \sim i} \sum_{k'} \mu(Y', k') &= \sum_{Y' \sim i, k'} \frac{\lambda^{Y'} Q_{k'}(Y')}{1 - S_{k'}(\lambda)} \\
&= \sum_{k'} \frac{G_i(Q_{k'}, \lambda)}{1 - S_{k'}(\lambda)} \\
&\leq \lambda_i - 1 \quad \text{by the hypothesis of Theorem 3.3(a).}
\end{aligned}$$

To prove the second claim: we have

$$\begin{aligned}
\sum_{Y' \bowtie_k Y} \mu(Y', k) &= \sum_{Y' \bowtie_k Y} \frac{\lambda^{Y'} Q_k(Y')}{1 - S_k(\lambda)} \\
&\leq \frac{S_k(\lambda)}{1 - S_k(\lambda)} \quad \text{as } Q_k(Y, k) > 0
\end{aligned}$$

□

It is useful to define

$$H_{i,j} = \sum_k \frac{G_{i,j}(Q_k, \lambda)}{1 - S_k(\lambda)}$$

and similarly the “summation notation” $H_i = \sum_j H_{i,j}$. Then the criterion of Theorem 3.3 has the simple form $\lambda_i - H_i \geq 1$. We may assume without loss of generality that $\lambda_{i,j} \geq H_{i,j}$ for each (i, j) ; for if not, we may set $\lambda_{i,j} = H_{i,j} = 0$ and still satisfy Theorem 3.3.

The parameter $H_{i,j}$ will turn out to play the crucial role in Section 3.1 which analyzes the LLL distribution.

3.1. The LLL distribution. If we are given weighting functions satisfying Theorems 2.6 or 2.6, then we know that there exists a configuration which avoids all bad events. Furthermore, such a configuration can be found by running the PRA. We may wish to learn more about such configurations, other than that they exist. We can use the probabilistic method, by defining an appropriate distribution on the set of feasible configurations. The PRA naturally defines a probability distribution, namely, the distribution imposed on elements after the algorithm terminates.

Suppose we are given some atomic event $E = \{(i_1, j_1), \dots, (i_r, j_r)\}$. We give the following definition:

Definition 3.5. (A strict neighbor-set for E) Given any $E \subseteq \mathcal{X}$ and a set $\mathcal{T} \subseteq 2^{\mathcal{X}}$, we say that \mathcal{T} is a strict neighbor-set for E if the following conditions hold:

- (1) For all $Z \in \mathcal{T}$ we have $Z \sim E$.
- (2) There do not exist $Z, Z' \in \mathcal{T}$ with $Z \sim Z'$

We can give the following bound on the output of the PRA. The proof is nearly identical to that in [9]:

Theorem 3.6. Suppose that μ satisfies Theorem 2.6(a). Then, for any atomic event E , the probability that E is true in the output of the PRA, is at most

$$P(\text{PRA output satisfies } E) \leq p^E \sum_{\substack{\mathcal{T} \text{ a strict} \\ \text{neighbor-set of } E}} \prod_{Y \in \mathcal{T}} \mu(Y) \leq p^E \prod_{Y \sim E} (1 + \mu(Y))$$

where recall we define $\mu(Y) = \sum_k \mu(Y, k)$.

Proof. We build a witness tree for the first time that E is true (if ever). This tree has a root node labeled E , which contributes in probability p^E . The labels of its children form a strict neighbor-set for E . We can show that the Witness Tree Lemma holds also for this type of witness tree, and that the sum of the weights of all such witness trees is at most $p^E \sum_{\substack{\mathcal{T} \text{ a strict} \\ \text{neighbor-set of } E}} \prod_{Y \in \mathcal{T}} \mu(Y)$ \square

We can easily reformulate this in terms of the our vector λ as well:

Theorem 3.7. Suppose that λ is a vector satisfying Theorem 3.3. Let E be some atomic event. The probability that E is ever true during the execution of the PRA is at most:

$$P(E \text{ true during the PRA}) \leq \prod_{(i,j) \in E} \lambda_{i,j} = \lambda^E$$

Proof. We apply Theorem 3.6. Suppose we enumerate \mathcal{T} , the strict neighbor-sets of E . For each $(i, j) \in E$, then \mathcal{T} may contain one or zero sets $Y \sim i$. For each such (i, j) , the total contribution of all such sets is bounded by: $1 + \sum_{Y \sim i} \mu(Y)$; as shown in Theorem 3.3, this is at most λ_i .

Thus, the probability that E is true is given by

$$\begin{aligned} P(\text{PRA output satisfies } E) &\leq p^E \sum_{\mathcal{T} \text{ a strict neighbor-set of } E} \prod_{Y \in \mathcal{T}} \mu(Y) \\ &\leq p^E \prod_{(i,j) \in E} \lambda_i \\ &= \prod_{(i,j) \in E} p_{ij} \lambda_i = \prod_{(i,j) \in E} \lambda_{ij} = \lambda^E \end{aligned}$$

\square

We can improve on Theorem 3.6 when the event E is defined by a *single variable*. In this case, the formulation in terms of a vector λ and Theorem 3.3 is particularly powerful. For these singleton events, we are even able to show a *lower bound* on the probability of E ; such lower bounds are not possible in general.

Theorem 3.8. *Suppose that λ satisfies Theorem 3.3. Let J be a set of assignments to variable i . The probability that the PRA terminates with $X_i \in J$ is upper-bounded by:*

$$P(X_i \in J) \leq \frac{\sum_{j \in J} \lambda_{i,j}}{\lambda_i - H_i + \sum_{j \in J} H_{i,j}}$$

Proof. In this proof, we use the notation (i, J) to denote an element (i, j) for $j \in J$ and similarly we use the notation (i, \bar{J}) to denote an element (i, j') for $j' \notin J$.

For any (Y, k) with $Y \ni (i, \bar{J})$, let $R(Y, k)$ denote the total weight of tree-structures rooted in (Y, k) , in which there is no node containing any $Y' \ni (i, J)$. Also define $R = \sum_{Y \ni (i, \bar{J}), k} R(Y, k)$. The first step in this proof is to bound R .

For any (Y, k) we enumerate such tree-structures as follows: the root may have a \bowtie_k -child of (Y, k) ; it may have children from any of the other categories in Y (other than i); it may have another child also rooted in some other (Y', k') where $Y' \ni (i, \bar{J})$.

By Proposition 3.4, for each $i' \neq i$, the total weight of all tree-structures rooted in (Y', k') for $Y' \sim i'$ is at most $\lambda_{i'} - 1$. Similarly the total weight of the \bowtie_k -children of (Y, k) is at most $\frac{S_k(\lambda)}{1 - S_k(\lambda)}$, and the total weight of tree-structures rooted in (i, J) is at most R .

So the contributions of these three terms are bounded by, respectively, $1 + \frac{S_k(\lambda)}{1 - S_k(\lambda)}$, $1 + R$ and $\prod_{i' \neq i, Y \sim i'} \lambda_{i'}$. We see thus that R satisfies the bound

$$(13) \quad R(Y, k) \leq \frac{p^Y Q_k(Y) (1 + R) \prod_{i' \neq i, i' \sim Y} \lambda_{i'}}{1 - S_k(\lambda)}$$

Summing over $Y \ni (i, \bar{J})$ and $k \in [K]$ we have

$$\begin{aligned} R &= \sum_{Y \ni (i, \bar{J}), k} R(Y, k) \\ &\leq \frac{\sum_{Y \ni (i, \bar{J}), k} p^Y Q_k(Y) (1 + R) \prod_{i' \neq i, i' \sim Y} \lambda_{i'}}{1 - S_k(\lambda)} \\ &= (1 + R) \sum_k \frac{\sum_{j \notin J} \sum_{Y \ni (i, j)} Q_k(Y) \lambda^Y \frac{p_{i,j}}{\lambda_{i,j}}}{1 - S_k(\lambda)} \\ &= (1 + R) \sum_{j \notin J} \frac{p_{i,j}}{\lambda_{i,j}} \sum_k \frac{G_{i,j}(Q_k, \lambda)}{1 - S_k(\lambda)} \\ &= \frac{1 + R}{\lambda_i} \sum_{j \notin J} H_{i,j} \end{aligned}$$

As $x/(1+x)$ is an increasing function of x , and we are assuming that $H_i \leq \lambda_i$, this implies that we have

$$\begin{aligned} R &\leq \frac{\sum_{j \notin J} \frac{H_{i,j}}{\lambda_i}}{1 - \sum_{j \notin J} \frac{H_{i,j}}{\lambda_i}} \\ &= \frac{H_{i, \bar{J}}}{\lambda_i - H_{i, \bar{J}}} \end{aligned}$$

Now that we have bounded R , we consider bounding the probability that $X_i \in J$. We may build a witness tree for the first time that we have $X_i \in J$ during the execution of the PRA. This tree contains a root node labeled (i, J) . It has either no children, or it has a single child node labeled (Y, k) where $Y \ni (i, \bar{J})$. Below the root node, this tree cannot contain any instances of (i, J) . Thus, the total weight of all such tree structures is at most $p_{i,J}(1 + R)$, which we can bound as:

$$\begin{aligned} p_{i,J}(1 + R) &\leq \sum_{j \in J} p_{i,j} \left(1 + \frac{H_{i,\bar{J}}}{\lambda_i - H_{i,\bar{J}}}\right) \\ &= \frac{\sum_{j \in J} p_{i,j} \lambda_i}{\lambda_i - H_{i,\bar{J}}} = \frac{\sum_{j \in J} \lambda_{i,j}}{\lambda_i - H_i + \sum_{j \in J} H_{i,j}} \end{aligned}$$

One can show that the Witness Tree Lemma holds for such tree-structures, and so this also bounds the total probability that $X_i \in J$. \square

A simple corollary of Theorem 3.8 shows a *lower bound* on the probability that the PRA terminates with $X_i \in J$:

Corollary 3.9. *Suppose that λ satisfies Theorem 3.3. Let J be a set of possible assignments to variable i . The probability that at the termination of the PRA we have $X_i \in J$ is at least:*

$$P(\text{PRA terminates with } X_i \in J) \geq \frac{\sum_{j \in J} \lambda_{i,j} - \sum_{j \in J} H_{i,j}}{\lambda_i - \sum_{j \in J} H_{i,j}}$$

Proof. Apply Theorem 3.8 to bound from above the probability of ever selecting (i, \bar{J}) . \square

4. BAD-EVENTS DEFINED BY SUMS OF RANDOM VARIABLES

In this section, we explore a connection between symmetric polynomials and tail bounds for sums of independent random variables. When the bad-events are defined in terms of such sums, then these symmetric polynomials correspond in a natural way with fractional hitting sets. Much of this section is based on [29], which drew the connection between symmetric polynomials and Chernoff bounds.

We begin by recalling two useful result of [29] bounding the value of multivariate symmetric polynomials.

Proposition 4.1. *For any real numbers $a_1, \dots, a_\ell \in [0, 1]$ and integer $k \geq 0$, we have that*

$$\sum_{\substack{X \subseteq [\ell] \\ |X|=k}} \prod_{x \in X} a_x \geq \binom{a_1 + \dots + a_\ell}{k}$$

Proposition 4.2. *For any real numbers $a_1, \dots, a_\ell \in [0, \infty)$ and integer $k \geq 0$, we have that*

$$\sum_{\substack{X \subseteq [\ell] \\ |X|=k}} \prod_{x \in X} a_x \leq \frac{(a_1 + \dots + a_\ell)^k}{k!}$$

We now introduce a particularly useful type of fractional hitting set for bad-events defined by sums of indicator variables for elements.

Theorem 4.3. *Suppose we are given an assignment of non-negative real numbers $\lambda_x \in \mathbf{R}_+$ and coefficients $a_x \in [0, 1]$ for each element $x \in \mathcal{X}$. (Recall that an element x refers to an ordered pair (i, j) , where $i \in [n]$ and j is a possible assignment to variable X_i)*

Define $\mu = \sum_x a_x \lambda_x$, and for each $i \in [n]$ define $\mu_i = \sum_j a_{i,j} \lambda_{i,j}$.

Suppose that there is a complex bad event \mathcal{B} defined by

$$\mathcal{B} \equiv \sum_x a_k[X_i = j] \geq t$$

where we use the Iverson notation for whether variable i takes on value j , where $t \geq \mu$. Let d be any non-negative integer. Then, recalling Definition 3.1, there is a fractional hitting-set Q with the property

$$G(Q, \lambda) \leq \frac{\mu^d}{d! \binom{t}{d}}; \quad G_i(Q, \lambda) \leq \frac{d\mu_i}{\mu} \cdot \frac{\mu^d}{d! \binom{t}{d}}.$$

We refer to the parameter d as the width of this hitting-set.

Proof. We define Q as follows: suppose that $Y \subseteq \mathcal{X}$. If $|Y| = d$, and Y does not contain two items from the same category, then set

$$Q(Y) = \frac{\prod_{x \in Y} a_x}{\binom{t}{d}} = \frac{a^Y}{\binom{t}{d}};$$

otherwise we set $Q(Y) = 0$. To show that Q is a valid fractional hitting set: suppose that B is an atomic bad-event; that is, we view B as a subset of \mathcal{X} with $\sum_{x \in B} a_x \geq t$. We may assume that B does not contain two items from the same category.

Then we have that

$$\begin{aligned} \sum_{Y \subseteq B} Q(Y) &= \frac{\sum_{Y \subseteq B, |Y|=d} a^Y}{\binom{t}{d}} \\ &\geq \frac{\binom{\sum_{x \in B} a_x}{d}}{\binom{t}{d}} \quad \text{by Proposition 4.1} \\ &\geq \frac{\binom{t}{d}}{\binom{t}{d}} = 1 \end{aligned}$$

as desired.

We can compute $G(Q, \lambda)$ as:

$$\begin{aligned} G(Q, \lambda) &= \sum_{Y \subseteq \mathcal{X}} \lambda^Y Q(Y) \\ &= \frac{\sum_{Y \subseteq \mathcal{X}, |Y|=d} \lambda^Y a^Y}{\binom{t}{d}} \\ &\leq \frac{(\sum_{x \in Y} a_x \lambda_x)^d}{d! \binom{t}{d}} \quad \text{by Proposition 4.2} \\ &= \frac{\mu^d}{d! \binom{t}{d}}. \end{aligned}$$

For $i \in [n]$, we similarly have

$$\begin{aligned} G_i(Q, \lambda) &= \sum_{Y \ni (i, j) \text{ for some } j} \lambda^Y Q(Y) \\ &= \frac{\sum_j a_{i,j} \lambda_{i,j} \sum_{\substack{Y \subseteq \mathcal{X} \\ |Y|=d-1 \\ Y \text{ contains no elements from category } i}} \lambda^Y}{\binom{t}{d}} \end{aligned}$$

$$\begin{aligned}
&\leq \frac{\sum_j a_{i,j} \lambda_{i,j} \left(\sum_{\substack{x \in \mathcal{X} \\ x \text{ is not from category } i}} a_x \lambda_x \right)^{d-1}}{(d-1)! \binom{t}{d}} && \text{by Proposition 4.2} \\
&= \frac{\mu_i (\mu - \mu_i)^{d-1}}{(d-1)! \binom{t}{d}} \\
&\leq \frac{d\mu_i}{\mu} \frac{\mu^d}{d! \binom{t}{d}}
\end{aligned}$$

□

Algorithmic aspects of Theorem 4.3. To implement the PRA using the fractional hitting set Q of Theorem 4.3, we must be able to efficiently access Q , in the following sense. Given a configuration X and an atomic bad-event B which is true on X , one must be able to efficiently select some $Y \subseteq B$ with probability proportional to $Q(Y)$. There may be up to $\binom{n}{d}$ such subsets Y , and so implemented naively this might take time n^d , potentially exponential time.

However, it is possible to perform this selection process more efficiently, as follows. Let B be an atomic event on items x_1, \dots, x_k with weights a_1, \dots, a_k . For any set $W \subseteq B$, define

$$R(W) = \sum_{\substack{Y: W \subseteq Y \subseteq B \\ |Y|=d}} Q(Y)$$

This can be evaluated in time $O(dk)$ using a dynamic program. Now, to sample $Y \subseteq B$ with probability proportional to $Q(Y)$, we use the following procedure:

1. Let $Y_0 = \emptyset$
2. For $i = 1, \dots, d$:
 3. For $j \in B - Y_{i-1}$ compute $q_j = R(Y_{i-1} \cup \{j\})$.
 4. Select some $j \in B - Y_{i-1}$ with probability proportional to q_j .
 5. Set $Y_i = Y_{i-1} \cup \{j\}$.
6. Return Y_d

Proposition 4.4. *Let $Z \subseteq B$ with $|Z| = d$. Then*

$$P(Y_d = Z) = \frac{Q(Z)}{\sum_{Y \subseteq B} Q(Y)}$$

Proof. We show by induction on i the following: for any sets $W \subseteq Z \subseteq B$ with $|W| = i, |Z| = d$, and $0 \leq i \leq d$, we have

$$P(Y_d = Z \mid Y_i = W) = \frac{Q(Z)}{R(W)}$$

Applying this with $i = 0, W = \emptyset$ will give us the desired result. Also, the induction case with $i = d$ is trivially true. For the induction step:

$$\begin{aligned}
P(Y_d = Z \mid Y_i = W) &= \frac{\sum_{z \in Z-W} R(W \cup \{z\}) P(Y_d = Z \mid Y_{i+1} = W + z)}{\sum_{x \in B-W} R(W \cup \{x\})} \\
&= \frac{\sum_{z \in Z-W} R(W \cup \{z\}) Q(Z) / R(W + z)}{\sum_{x \in B-W} R(W \cup \{x\})} \quad \text{induction hypothesis} \\
&= \frac{Q(Z)(d-i)}{\sum_{x \in B-W} \sum_{Y: W \cup \{x\} \subseteq Y \subseteq Z} Q(Y)} \\
&= \frac{Q(Z)(d-i)}{\sum_{Y: W \subseteq Y \subseteq Z} \sum_{x \in Y-W} Q(Y)} \\
&= \frac{Q(Z)(d-i)}{R(W)(d-i)} = \frac{Q(Z)}{R(W)}
\end{aligned}$$

thus completing the induction. \square

4.1. Chernoff Bounds. There is a close connection between the Chernoff bounds, symmetric polynomials, and the fractional hitting set defined in Theorem 4.3. To state this in its clearest form, we define the Chernoff upper-tail separation function; this will play a key role in our results.

Definition 4.5. (The Chernoff upper-tail separation function) For $0 < \mu \leq t$, letting $\delta = \delta(\mu, t) = t/\mu - 1 \geq 0$, we define

$$Ch(\mu, t) = e^{t-\mu} (\mu/t)^t$$

i.e., $Ch(\mu, t)$ is the Chernoff bound that a sum of $[0, 1]$ -bounded and independent random variables with mean μ will exceed t . If $t < \mu$ we define $Ch(\mu, t) = 1$.

It is often convenient to state the Chernoff bounds in terms of the relative deviation between the threshold t and the mean μ . In such cases, we define $\lambda = t/\mu - 1$. We can then restate the Chernoff bound as

$$Ch(\mu, \mu(1 + \lambda)) = \left(\frac{e^\lambda}{(1 + \lambda)^{1+\lambda}} \right)^\mu$$

The following is a key connection between Chernoff bounds and Theorem 4.3:

Theorem 4.6 ([29]). Let $0 \leq \mu \leq t$. Then for $d = \lceil t - \mu \rceil$, we have

$$\frac{\mu^d}{d! \binom{t}{d}} \leq Ch(\mu, t)$$

However, we note some key differences between our use of Theorem 4.6, and its use by [29]. It is shown by [29] that the choice of $d = \lceil t - \mu \rceil$ minimizes the quantity $\frac{\mu^d}{d! \binom{t}{d}}$; thus, in terms of approximating Chernoff bounds via symmetric polynomials, this choice of d is *optimal*.

In our context, however, there are many parameters related to the PRA which depend on d ; the choice of $d = \lceil t - \mu \rceil$ is often a good choice, but is not necessarily the best for the overall algorithm.

In addition, it is often possible to achieve substantially better bounds using Theorem 4.3 than are available through the Chernoff bound. This is particularly the case when μ, t are fixed finite quantities. This will come up in numerous places in Section 7, where we carefully optimize d .

5. TRANSVERSALS WITH OMITTED SUBGRAPHS

Suppose we are given a graph $G = (V, E)$ with a partition of its vertices into sets $V = V_1 \sqcup V_2 \sqcup \dots \sqcup V_l$, each of size b . We refer to these sets as *blocks* or *classes*. We wish to select exactly one vertex from each block. Such a set of vertices $A \subseteq V$ is known as a *transversal*. There is a large literature on selecting transversals such that the graph induced on A omits certain subgraphs. (This problem was introduced in a slightly varying form by [6]; more recently it has been analyzed in [31, 15, 32, 16, 13]). For example, when A is an independent set of G (omits the 2-clique K_2), this is referred to as an *independent transversal*.

It is well-known that a graph G with n vertices and average degree d has an independent set of size at least $n/(d+1)$. For an independent transversal, a similar criterion exists. Alon gives a short LLL-based proof that a sufficient condition for such an independent transversal to exist is to require $b \geq 2e\Delta$ [2], where Δ is the maximum degree of any vertex in the graph. Haxell provides an elegant topological proof that a sufficient condition is $b \geq 2\Delta$ [14]. The condition of [14] is existentially optimal, in the sense that $b \geq 2\Delta - 1$ is not always admissible [16, 32, 31]. The work of [15] gives a similar criterion of $b \geq \Delta + \lfloor \Delta/r \rfloor$ for the existence of a transversal which induces no connected component of size $> r$. (Here $r = 1$ corresponds to independent transversals.) Finally, the work of [22] gives a criterion of $b \geq \Delta$ for the existence of a transversal omitting K_3 ; this is the optimal constant but the result is non-constructive.

These bounds are all given in terms of the *maximum degree* Δ , which can be a crude statistic. The proof of [14] adds vertices one-by-one to partial transversals, which depends very heavily on bounding the maximum degree of any vertex. It is also highly non-constructive. Suppose we let d denote the maximum *average* degree of any class V_i (that is, we take the average of the degree (in G) of all vertices in V_i , and then maximize this over all i). This is a more flexible statistic than Δ . We present our first result here in parts (R1), (R2), and (R3) of Theorem 5.1. As shown in [16, 32, 31], the result of (R1) cannot be improved to $b \geq 2\Delta - 1$ (and hence, in particular, to $b \geq 2d - 1$). As shown in [22], the result of (R3) cannot be improved to $b \geq cd$ for any constant $c < 1$. The result (R1) for independent transversals can also be obtained using the LLL variant of [24], but (R2) and (R3) appear new to our knowledge.

Theorem 5.1. *Suppose we have a graph G whose vertex set is partitioned into blocks of size at least b . Suppose that the average degree of the vertices in each block is at most d . Then:*

- (R1): *If $b \geq 4d$, then G has an independent transversal;*
- (R2): *If $b \geq 2d$, then G has a transversal which induces no connected component of size > 2 ;*
- (R3): *If $b \geq (4/3)d$, then G has a transversal which induces no 3-clique K_3 .*

Furthermore, these transversals can be constructed in expected polynomial time.

Proof. We define l separate variables X_1, \dots, X_l ; variable X_l selects which of the b vertices in block l go into the transversal. For each forbidden subgraph which appears in G , we associate an atomic bad event. (Note that the atomic bad events for (R2) are all the path of length two in G ; for (R3) they are the triangles of G .)

We use the trivial labeling $K = 1$. Each forbidden subgraph corresponds to an atomic bad-event. We define a fractional hitting set Q as follows. For each edge $f = \langle u, v \rangle \in E$ we assign weight $B'(\{u, v\}) = 1/r$, where r is a parameter depending on the structure we are avoiding. For case (R1), we assign $r = 1$. For case (R2), we assign $r = 2$; for case (R3) we assign $r = 3$. (B' is zero everywhere else.) Now, note that in any of the three cases, the atomic bad events all involve exactly r edges, so the fractional hitting set is valid. Furthermore, any pair of such edges overlap in at least one vertex, so the \bowtie relation is null in this case.

Note that the precondition of Theorem 3.3(c) holds here. We apply Theorem 3.3(c) with all entries of λ being α , for some scalar α to be determined. Let d_v denote the degree of vertex v .

Then, in order to prove $\lambda_i \geq 1 + \sum_k G_i(Q_k, \lambda)$, we need to show

$$b\alpha - \sum_{v \in V_i} d_v \alpha^2 / r \geq 1, \text{ i.e., } b\alpha - bd\alpha^2 / r \geq 1 \text{ suffices.}$$

This has a solution $\alpha > 0$ iff $b \geq \frac{4d}{r}$, which gives us the three claimed results. \square

5.1. Avoiding large cliques. For avoiding cliques of size $s > 3$, the above approach based on the maximum average degree d no longer works; we instead give a bound in terms of the maximum degree Δ . We will be interested in the case when both s and Δ is large. More specifically, for any value of s we seek a bound of the form $b \geq \gamma_s \Delta$ (where b, Δ may go to infinity). We then seek to understand the behavior of the value of γ_s as $s \rightarrow \infty$.

We must have $\gamma_s \geq 1/(s-1)$. To see this, note that for any value of $b \geq 1$, we may take a graph consisting of s blocks each containing b vertices, where every vertex is connected to all vertices outside its block. This graph has $\Delta = b(s-1)$, and clearly any transversal contains a copy of K_s . An argument of [31] shows the slightly stronger lower bound $\gamma_s \geq \frac{s}{(s-1)^2}$; intriguingly, this is conjectured in [31] to be exactly tight. On the other hand, a construction in [22] shows that $\gamma_s \leq 2/(s-1)$. This is non-constructive, even for fixed s ; this is the best general upper-bound on γ_s previously known.

We show that the lower-bound of [31] gives the correct *asymptotic* rate of growth, up to lower-order terms; i.e., we show in Theorem 5.2 that $\gamma_s \leq 1/s + o(1/s)$. In fact, we will show that when $b \geq \Delta/s(1 + o(1/s))$, that we can find a transversal which avoids any s -star; that is, all vertices have degree $< s$. This implies that the transversal avoids K_s . Furthermore, such transversals can be found in polynomial time.

Comparison with the standard LLL: Before we give our construction based on the PRA, we discuss how one might approach this problem using the standard LLL, and why this approach falls short. As in [2], we make the natural random choice for a transversal: choose a vertex randomly and independently from each V_i . Suppose we define, for each s -clique H of the graph, a separate bad event. Each bad event has probability $(1/b)^s$. We calculate the dependency of an s -clique as follows: for each vertex $v \in H$, we choose another v' in the block of v , and v' may be involved in up to $\Delta^{s-1}/(s-1)!$ other s -cliques. This gives us the LLL criterion

$$e \times (1/b)^s \times sb\Delta^{s-1}/(s-1)! \leq 1$$

which gives us the criterion

$$b/\Delta \geq \left(\frac{es}{(s-1)!} \right)^{\frac{1}{s-1}} = e/s + o(1/s)$$

In implementing the PRA, it is simpler to enforce the stronger condition that the traversal omits s -stars. (That is, in the traversal T , no vertex may have induced degree $\geq s$). We will resample the central vertex as well as $r \leq s$ of the exterior vertices in an s -star, chosen uniformly. We thus obtain our result:

Theorem 5.2. *There is a constant $c > 0$ such that, whenever*

$$b \geq \frac{\Delta}{s} \left(1 + c \sqrt{\frac{\log s}{s}} \right)$$

then there is a transversal which omits any s -stars. Furthermore, such a transversal can be found in polynomial time.

In particular, we have

$$\gamma_s \leq \frac{1}{s} \left(1 + O\left(\sqrt{\frac{\log s}{s}} \right) \right)$$

Proof. We will use Theorem 3.3(c), assigning the constant vector $\vec{\lambda} = \alpha$ where α is a constant to be chosen. We use the trivial labeling. We use the following fractional hitting set: for each H consisting of a single vertex and r neighbors, we assign weight $\binom{s}{r}^{-1}$; for all other sets we assign weight zero. In this case, \bowtie is null: any two r -stars H, H' which both correspond to the same s -star, will overlap in their central vertex.

For any vertex v , there are $\leq \binom{\Delta}{r}$ r -stars in which v is the central vertex and $\leq \Delta \binom{\Delta-1}{r-1}$ r -stars where it is a peripheral vertex. So the condition of Theorem 3.3(c) becomes

$$b\alpha - b\left(\binom{\Delta}{r} + \Delta\binom{\Delta-1}{r-1}\right)\binom{s}{r}^{-1}\alpha^{r+1} \geq 1$$

We estimate this as:

$$\begin{aligned} b\alpha - b\left(\binom{\Delta}{r} + \Delta\binom{\Delta-1}{r-1}\right)\binom{s}{r}^{-1}\alpha^{r+1} &\geq b\alpha - b\left(\frac{\Delta^r}{r!} + \frac{\Delta^r}{(r-1)!}\right)\frac{r!(s-r)!}{s!}\alpha^{r+1} \\ &\geq b\alpha - b((r+1)\Delta^r\frac{(s-r)!}{s!})\alpha^{r+1} \\ &\geq b\alpha - b(r+1)\Delta^r(s-r)^{-r}\alpha^{r+1} \end{aligned}$$

Now set

$$\alpha = \frac{s-r}{\Delta(1+r)^{2/r}}$$

and the condition $b\alpha - b(r+1)\Delta^r(s-r)^{-r}\alpha^{r+1} \geq 1$ reduces to showing that

$$\frac{br(s-r)}{\Delta(1+r)^{1+2/r}} \geq 1$$

which is satisfied for

$$b \geq (\Delta/s) \times \frac{(r+1)^{1+2/r}}{r(1-r/s)}$$

At this point we set $r = \lceil \sqrt{s \ln s} \rceil$. As $(r+1)^{1+2/r}/r$ is a decreasing function of r , we have that:

$$\frac{(r+1)^{1+2/r}}{1-r/s} \leq \frac{(1+\sqrt{s \ln s})^{1+\frac{2}{\sqrt{s \ln s}}}}{\sqrt{s \ln s}(1-\frac{1+\sqrt{s \ln s}}{s})}$$

It may now be verified that we have $\frac{(1+\sqrt{s \ln s})^{1+\frac{2}{\sqrt{s \ln s}}}}{\sqrt{s \ln s}(1-\frac{1+\sqrt{s \ln s}}{s})} \leq 1 + O(\sqrt{\frac{\log s}{s}})$. This implies that the PRA converges under the criterion

$$b \geq (\Delta/s)(1 + c\sqrt{\frac{\log s}{s}})$$

for some sufficiently large constant $c > 0$.

To implement a step of the PRA, one must search the graph for any s -star in the current candidate transversal; this can be done easily in polynomial time. \square

We reiterate that this result improves on [22] in three distinct ways: it gives a better asymptotic estimate for γ_s ; it is fully constructive; it finds a transversal omitting not only s -cliques but also s -stars.

5.2. Weighted independent transversals. We next study *weighted* transversals, as considered by [1]. We use our weighting condition to lower- and upper- bound the weights of independent transversals, which is not possible using [14] or [5].

Suppose that we are given weights $w(v) \geq 0$ for each vertex of G . There is a simple argument that $G = (V, E)$ has an independent set of weight at least $\frac{w(V)}{\Delta+1}$ and that G has a transversal (not necessarily independent) of weight at least $\frac{w(V)}{b}$. Likewise, G has independent sets or transversals with weight at most $w(V)/(\Delta+1)$ or $w(V)/b$, respectively. Note also that we cannot expect independent transversals of weight more (or less) than $w(V)/b$ in general: e.g., consider the case of all weights being equal. Our theorems 5.3 and 5.4 improve quite a bit upon the (nonconstructive) bounds of [10]; among other things, we show next that weight at least $\frac{w(V)}{b}$ is in fact achievable if $b \geq 4.5\Delta$, a result that was shown to be true asymptotically for large b in [10].

Theorem 5.3. *Suppose $4\Delta \leq b \leq 4.5\Delta$. Then there is an independent transversal $I \subseteq V$ with weight*

$$w(I) \geq w(V) \left(\frac{\sqrt{b} + \sqrt{b-4\Delta}}{\sqrt{b}(2b-1) + \sqrt{b-4\Delta}} \right) \geq \frac{w(V)}{8\Delta-1}.$$

Suppose $b \geq 4.5\Delta$. Then there is an independent transversal $I \subseteq V$ with weight

$$w(I) \geq w(V) \cdot \min\left(\frac{1}{b}, \frac{4}{27\Delta-2}\right).$$

Furthermore, independent transversals with weight at least $(1-n^{-\Theta(1)})$ times these lower-bounds, can be found in polynomial time with high probability.

Proof. The first result follows in a straightforward manner from Theorem 3.8 when we set each entry of λ to the scalar constant $\alpha = \frac{b-\sqrt{b}\sqrt{b-4\Delta}}{2b\Delta}$; we use a single bad-event B_1 with the trivial hitting-set, whose atomic events correspond to every edge separately. Then the \bowtie_1 -relation is null and we have $\hat{S}_1 = 1$, and the condition of Theorem 2.6(c) requires

$$b\alpha - b\Delta\alpha^2 \geq 1$$

which is easily checked. Now for each vertex $v \in V_i$ we have $H_{i,v} \leq \alpha^2\Delta$; so the probability of selecting this vertex in the LLL distribution is

$$P(\text{select } v) \geq \frac{\lambda_{i,j} - H_{i,v}}{\lambda_i - H_{i,v}} \geq \frac{\alpha - \alpha^2\Delta}{b\alpha - \alpha^2\Delta} = \frac{\sqrt{b} + \sqrt{b-4\Delta}}{\sqrt{b}(2b-1) + \sqrt{b-4\Delta}}$$

For the second result, we assume that the vertices are sorted in decreasing order of weight; for any block i , we write the vertices as $v_{i,1}, v_{i,2}, \dots, v_{i,b}$ where $w(v_{i,1}) \geq w(v_{i,2}) \geq \dots \geq w(v_{i,b})$. Our strategy now will be to set $\vec{\lambda}_{i,j} = \frac{1}{3\Delta}$ for $1 \leq j \leq r = \lfloor 9\Delta/2 \rfloor$, and set $\vec{\lambda}_{i,j} = 0$ for $j > r$. That is, we only assign positive probability to the r -highest-weight vertices in each block.

We now have for each block i

$$\sum_j \lambda_{i,j} - \sum_j \sum_{\text{edges } f \text{ involving } v_{i,j}} \alpha^2 \geq r\alpha - r\Delta\alpha^2 \geq 1$$

so the PRA criterion is satisfied.

We will now compute the expected weight of the resulting solution. Let us fix now a block V_i , and X_i is the resulting variable which is the selected vertex in that block. We write $w_j = w(v_{i,j})/w(V_i)$ for each $j \leq b$.

As we have $X_i \leq r$ with probability one, we have

$$w(V_i \cap I) = w(V_i) \left(w_r + \sum_{j=1}^r [X_i = j](w_j - w_r) \right)$$

using the Iverson notation. Observe that we have $w_j \geq w_r$ for all $j \leq r$. By Theorem 3.9, we have

$$P(X_i = j) \geq \frac{\alpha - \alpha^2 \Delta}{r\alpha - \alpha^2 \Delta} = \frac{4}{27\Delta - 2}$$

Let us write $q = \frac{4}{27\Delta - 2}$ and $t = \sum_{j=1}^r w_j$. We must have $r/b \leq t \leq 1$ and we also must have $w_r \geq \frac{1-t}{b-r}$ because the vertices are in sorted order.

We now have:

$$\begin{aligned} \mathbf{E}[w(V_i \cap I)] &\geq w(V_i) \left(w_r + q(t - rw_r) \right) \\ &= w(V_i) \left(w_r(1 - rq) + qt \right) \\ &\geq w(V_i) \left(\frac{1-t}{b-r}(1 - rq) + qt \right) \quad \text{as } rq \leq 1 \\ &= w(V_i) \left(\frac{1}{b-r}(1 - rq) + t \left(q - \frac{1-rq}{b-r} \right) \right) \end{aligned}$$

Suppose now that $bq \geq 1$. Then we estimate this using a lower bound for t :

$$\mathbf{E}[w(V_i \cap I)] \geq w(V_i) \left(\frac{1}{b-r}(1 - rq) + (r/b) \frac{bq - 1}{b-r} \right) = \frac{w(V_i)}{b}$$

On the other hand, if $bq \leq 1$, we estimate this expression using the upper bound for t :

$$\mathbf{E}[w(V_i \cap I)] \geq w(V_i) \left(\frac{1}{b-r}(1 - rq) + \frac{bq - 1}{b-r} \right) = qw(V_i)$$

Either way, we see that we have $\mathbf{E}[w(V_i \cap I)] \geq w(V_i) \min(q, 1/b) = w(V_i) \min(\frac{1}{b}, \frac{4}{27\Delta - 2})$ as desired.

Finally, the high-probability bound follows by standard repetition of this basic randomized algorithm. \square

We show a matching upper bound on weights:

Theorem 5.4. *Suppose $4\Delta \leq b \leq 8\Delta$. Then there is an independent transversal $I \subseteq V$ with weight*

$$w(I) \leq w(V) \frac{2}{b + 4\sqrt{(b - 4\Delta)\Delta}}$$

Suppose $b \geq 8\Delta$. Then there is an independent transversal $I \subseteq V$ with weight

$$w(I) \leq \frac{w(V)}{b}$$

Furthermore, independent transversals with weight at most $(1 + n^{-\Theta(1)})$ times these upper-bounds, can be found in polynomial time with high probability.

Proof. We sort the vertices in increasing order of weight, so that in each block i we have $w(v_{i,1}) \leq w(v_{i,2}) \leq \dots \leq w(v_{i,b})$. We define our $\vec{\lambda}$ by $\lambda_{i,j} = \alpha = \frac{1}{2\Delta}$ for $j \leq r = 4\Delta$ and $\lambda_{i,j} = 0$ for $j > r$. We have a single bad-event with trivial hitting-set and atomic bad-events for each edge. Then we have $\hat{S}_1 = 1$, and the condition of Theorem 3.3(c) is satisfied.

Fix a block i , and suppose the PRA chooses vertex X_i in that block. Write $w_j = w(v_{i,j})/w(V_i)$ for $j \leq b$.

We have

$$w(V_i \cap I) = w(V_i) \left(w_1 + (w_2 - w_1)[X_i \geq 2] + (w_3 - w_2)[X_i \geq 3] + \dots + (w_r - w_{r-1})[X_i = r] \right)$$

Now for each vertex $v \in V_i$ we have $H_{i,v} \leq \alpha^2 \Delta \leq \frac{1}{4\Delta}$. So for any $j \geq 1$, by Theorem 3.8 we have $P(X_i \geq j) \leq \frac{(r-j+1)\alpha}{r\alpha - (j-1)\frac{1}{4\Delta}} = \frac{2(r-j+1)}{2r-j+1}$. We write $q_j = \frac{2(r-j+1)}{2r-j+1}$ for the remainder of this proof.

So we have that

$$\begin{aligned}
\mathbf{E}[w(V_i \cap I)] &= w(V_i) \left(w_1 + (w_2 - w_1)P(X_i \geq 2) + (w_3 - w_2)P(X_i \geq 3) + \cdots + (w_r - w_{r-1})P(X_i \geq r) \right) \\
&\leq w(V_i) \left(w_1 q_1 + (w_2 - w_1)q_2 + (w_3 - w_2)q_3 + \cdots + (w_r - w_{r-1})q_r \right) \\
&= w(V_i) \left(w_1(q_1 - q_2) + w_2(q_2 - q_3) + \cdots + w_r q_r \right) \\
&= w(V_i) \sum_{j=1}^r 2r \frac{w_j}{(2r-j)(2r-j+1)}
\end{aligned}$$

In order to achieve an upper bound, we now consider *maximizing* the linear function $F(u_1, \dots, u_b) = 2r \sum_{j=1}^r \frac{u_j}{(2r-j)(2r-j+1)}$ subject to the constraints $u_1 \leq u_2 \leq \dots \leq u_b$ and $u_1 + u_2 + \dots + u_b = 1$. As F is linear, it has a basic solution (i.e. a solution at a corner of the polytope) which maximizes it. We claim that any basic solution $\langle u_1, \dots, u_b \rangle$ which maximizes F must have the following form: there is some integer $0 \leq k \leq r-1$, with the property that $u_1 = u_2 = \dots = u_k = 0$ and $u_{k+1} = u_{k+2} = \dots = u_b$.

To see this, suppose we are given some vector $\langle u_1, \dots, u_b \rangle$ which maximizes $F(u_1, \dots, u_b)$. We claim first that we must have $u_{r+1} = u_{r+2} = \dots = u_b = u_r$. For, suppose k is minimal such that $u_k > u_r$ strictly. Then we may increase u_r by ϵ and decrement u_k by ϵ and thereby increase $F(u)$ while preserving all constraints.

So we now may assume that $u_{r+1} = \dots = u_b = r$. Suppose that w does not have the indicated form; so let j denote the maximal index such that $u_j > 0$, and let k denote the maximal index such that $u_k < u_b$. If $j = k$, then $\langle u_1, \dots, u_b \rangle$ has the desired form.

So suppose $j < k$. As $u_r = u_{r+1} = \dots = u_b$, it must be that $k < r$. Then we may increment u_k by some small ϵ and decrement u_j by ϵ and thereby increase $F(\langle u_1, \dots, u_b \rangle)$; this preserves all the constraints.

So now we suppose that $j = k$. So $0 < u_k < u_{k+1}$ strictly. Then change u_k by some $+\epsilon$ and change u_{k+1}, \dots, u_b by $-\epsilon/(b-k)$; this preserves all constraints for ϵ in some small open ball around zero. This implies that u_1, \dots, u_b is not a basic solution.

In conclusion, we may assume that the maximum value of $F(u)$ is obtained some by $\langle u_1, \dots, u_b \rangle$ of the indicated form. Let $x = u_{k+1} = \dots = u_b$. As $u_1 + u_2 + \dots + u_b = 1$, it must be that $x = \frac{1}{b-k}$. Then we have that

$$\begin{aligned}
F(u) &= 2r \sum_{j=k+1}^r \frac{x}{(2r-j)(2r-j+1)} \\
&= 2rx \left(\frac{1}{r} - \frac{1}{2r-k} \right) \\
&= \frac{2(r-k)}{(b-k)(2r-k)}
\end{aligned}$$

This implies that, for any vector u satisfying the constraints $u_1 \leq u_2 \leq \dots \leq u_b$ and $u_1 + \dots + u_b = 1$, we have

$$\begin{aligned}
F(u) &\leq \max_{k \in \{0, \dots, r-1\}} \frac{2(r-k)}{(b-k)(2r-k)} \\
&\leq \max_{k \in [0, r-1]} \frac{2(r-k)}{(b-k)(2r-k)}
\end{aligned}$$

This can be regarded as a differentiable function of k ; the derivative has zeroes at $k = r \pm \sqrt{br - r^2}$. When $b \geq 8\Delta$, these zeroes are outside the range $[0, r-1]$ and $F(w)$ is a decreasing

function of k . So in that case, we upper bound it by its value at $k = 0$, namely

$$F(u) \leq 1/b$$

When $b < 8\Delta$, then one can see that the maximum value of $\frac{2(r-k)}{(b-k)(2r-k)}$ for $k \in [0, r-1]$ occurs at $k = r - \sqrt{br - r^2}$; at this point, we have

$$F(u) \leq \frac{2}{b + 2\sqrt{(b-r)r}} = \frac{2}{b + 4\sqrt{(b-4\Delta)\Delta}}$$

So we have that

$$\begin{aligned} \mathbf{E}[w(V_i \cap I)] &\leq w(V_i) \sum_{j=1}^r 2r \frac{w_j}{(2r-j)(2r-j+1)} \\ &= w(V_i) F(w) \\ &\leq w(V_i) \max\left(\frac{1}{b}, \frac{2}{b + 4\sqrt{(b-4\Delta)\Delta}}\right) \end{aligned}$$

as desired. \square

We can give similar bounds for independent transversals omitting other subgraphs. Note that such a bound cannot be specified in terms of the average degree, because we might add vertices of small degree and weight.

6. COLUMN-SPARSE ASSIGNMENT-PACKING PROBLEMS

In light of Theorem 4.3, consider the family of constraint satisfaction problems (CSPs) where we have a series of linear packing constraints of the form “ $\sum_{i,j} a_{k,i,j} [X_i = j] \leq b_k$ ”, where the n variables X_1, \dots, X_n which take their values from some subsets of the integers L_1, \dots, L_n respectively. We assume that $a_{k,x} \in [0, 1]$, $b_k \geq 0$. We will also assume that this CSP is “column-sparse”, in the sense that for any $x \in \mathcal{X}$ we have $\sum_k a_{k,x} \leq D$ for some parameter $D \geq 1$. (Here and in what follows, x will often refer to some element (i, j) .) When does such an integer linear program have a feasible solution?

Suppose we wish to solve this via LP relaxation. One technique is to solve the linear program, where we create fractional variables z_x for each element $x = (i, j)$; the variable z_x represents (fractionally) that $X_i = j$. We introduce a linear constraint $\forall i \sum_{j \in L_i} z_{i,j} = 1$. In addition the packing constraints are tightened to $\sum_x a_{k,x} z_x \leq c_k$ for some $c_k \leq b_k$. We then seek to use a randomized rounding to convert the fractional solution z_x into an integral solution.

We note that there are constructions using the standard LLL and standard MT algorithm that can yield results qualitatively similar to the ones in this section. The simplest case for these discrepancy results is when all the packing coefficients c_k have similar magnitudes; so suppose that for all k we have $b_k \leq R$; we compare our result with those of [12] and [19] in this regime.

The analysis of [19] is phrased in terms of the ℓ_0 -column norm D' , that is,

$$D' = \max_x \sum_k [a_{k,x} > 0]$$

As we require that $a_{k,x} \in [0, 1]$ we always have $D \leq D'$, and it possible that $D \ll D'$. It is quite difficult for the standard LLL to deal with fractional coefficients in the constraint matrix; the reason is that the variable X_i affects constraint k if $a_{k,i,j} > 0$, and it is possible that every variable affects every constraint.

The analysis of [12] shows how to use the ℓ_1 -norm D with the LLL. It constructs a solution with roughly $b_k = R + O(\sqrt{R \log(RD)})$, but it applies only to situations in which the fractional solution is given by $z_{i,1} = z_{i,2} = 1/2$. This is useful for problems based on hypergraph discrepancy and its generalizations, but is not useful if the fractional solution Z comes from some problem-specific LP

relaxation. We discuss this application in Section 6.3, in which we show that our framework can achieve the stronger bound $b_k = R + O(\sqrt{R \log D})$.

The case in which c_k have differing magnitudes appears to be beyond the scope of either of these analyses. In addition, these papers are quite difficult technically; in [12], there is a quantization argument, in which one must handle separately coefficients of different magnitudes. In [19], there is an iterated application of the LLL, in which one must track quite carefully the sizes of the relevant parameters as they are reduced from the original system. The PRA provides however, a simple and comprehensive framework to obtain an integral solution. A single application of the PRA directly produces our desired solution, and fractional coefficients are handled almost automatically.

6.1. LP rounding for column-sparse packing problems: the PRA. Our condition on the separation between b_k and c_k is based on the Chernoff bound. We will develop our algorithmic results in two parts. First, we state a general criterion in terms of the Chernoff upper-tail bounds; then, we give one possible solution to the resulting system of equations.

Theorem 6.1. *Suppose that we have an LP parametrized by $a_{k,x}, c_k$, where $D \geq \max_x \sum_{k,x} a_{k,x}$.*

Now let $\epsilon \in [0, 1]$, $b_k \in [1, \infty)^K$ satisfy the following conditions for all $k \in [K]$

$$\text{(C1): } \left(\frac{b_k+1}{c_k(1+\epsilon)} - 1 \right) \text{Ch}(c_k(1+\epsilon), b_k) \leq \frac{\epsilon}{4D}$$

$$\text{(C2): } \text{Ch}(c_k(1+\epsilon), b_k) \leq 1/2$$

Then if the linear program

$$\sum_{j \in L_i} z_{i,j} = 1, \quad \sum_x a_{k,x} z_x \leq c_k, \quad z_x \in [0, 1]$$

is satisfiable, then so is the integer program

$$X_i \in L_j \quad \sum_{i,j} a_{k,i,j} [X_i = j] \leq b_k$$

Furthermore, suppose we have a separation oracle for the LP and the IP. (That is, given a variable assignment, we can either find a violated linear constraint, or determine that all constraints are satisfied). Then such a satisfying assignment can be found in time which is polynomial in the number n of blocks.

Remark: *Because of our convention that $\text{Ch}(\mu, t) = 1$ for $t \leq \mu$, then (C2) automatically implies $b_k \geq c_k(1+\epsilon)$.*

Proof. Using the separation oracle, solve the LP in polynomial time. Suppose we have a feasible solution Z to the relaxed linear program. Then we set $\lambda = (1+\epsilon)Z$. We associate a distinct label to each packing constraint being violated. We will use the framework of Theorem 2.12, in which each packing constraint will correspond to its complex bad-event.

Let us analyze a constraint k ; this corresponds to the complex bad-event $\sum_{i,j} a_{k,i,j} [X_i = j] \geq b_k$. Let $r = (1+\epsilon)c_k$. So we may use the fractional hitting-set defined in Theorem 4.3, with $t = b_k$ and $d = \lceil t - r \rceil$. Using the terminology of Theorem 4.3, we have

$$\mu = \sum_{i,j \in L_i} a_{k,i,j} \lambda_{i,j} = \sum_{i,j \in L_i} a_{k,i,j} (1+\epsilon) z_{i,j} \leq (1+\epsilon) c_k = r$$

by (C2) this satisfies $\mu < t$.

We will next bound the contribution to G, G_i of each constraint k . By Theorem 4.3

$$G(Q_k, \lambda) \leq \frac{\mu^d}{d! \binom{t}{d}} \leq \frac{r^d}{d! \binom{t}{d}}.$$

As $d = \lceil t - r \rceil$, by Theorem 4.6 have $\frac{r^d}{d! \binom{t}{d}} \leq \text{Ch}(r, d)$; by (C2) this is $\leq 1/2$. So we have $G(Q_k, \lambda) \leq 1/2$.

Furthermore for each variable i we have

$$\begin{aligned}
G_i(Q_k, \lambda) &\leq \frac{\mu_i d}{\mu} \frac{\mu^d}{d! \binom{t}{d}} \\
&\leq \frac{\mu_i d}{r} \frac{r^d}{d! \binom{t}{d}} \\
&\leq \frac{\mu_i (t - r + 1)}{r} \text{Ch}(r, t) \quad \text{by Theorem 4.6} \\
&= \mu_i \left(\frac{b_k + 1}{c_k(1 + \epsilon)} - 1 \right) \text{Ch}(c_k(1 + \epsilon), b_k) \\
&\leq \mu_i \frac{\epsilon}{4D} \quad \text{by (C1)}
\end{aligned}$$

We compute μ_i by:

$$\mu_i = \sum_{j \in L_i} a_{k,i,j} \lambda_{k,i,j} \leq (1 + \epsilon) \sum_j a_{k,i,j} z_{i,j}$$

Now, we wish to show the PRA criterion is satisfied. We apply Theorem 3.3(b), so we sum over all possible assignment j obtaining

$$\begin{aligned}
\sum_{j \in L_i} \lambda_{i,j} - \sum_k \frac{G_i(Q_k, \lambda)}{1 - G(Q_k, \lambda)} &\geq (1 + \epsilon) - \sum_k \frac{\epsilon / (4D) \times (1 + \epsilon) \sum_j a_{k,i,j} z_{i,j}}{1/2} \\
&= (1 + \epsilon) - \frac{1}{2} \sum_{j \in L_i} z_{i,j} (1 + \epsilon) \times (\epsilon / D) \times \sum_k a_{k,i,j} \\
&\geq (1 + \epsilon) - \frac{1}{2} \sum_{j \in L_i} z_{i,j} (1 + \epsilon) \times (\epsilon / D) \times D \quad \text{definition of } D \\
&= (1 + \epsilon) - \frac{1}{2} \sum_{j \in L_i} z_{i,j} (1 + \epsilon) \epsilon \\
&= (1 + \epsilon) - \frac{1}{2} (1 + \epsilon) \epsilon \quad \text{as } \sum_{j \in L_i} z_{i,j} = 1 \\
&\geq 1 \quad \text{as } \epsilon \in [0, 1]
\end{aligned}$$

Furthermore, for any $i \in [n]$ we have $\sum_{j \in L_i} \lambda_{i,j} \leq 1 + \epsilon \leq 2$, so the expected number of iterations before the PRA terminates is at most $\sum_{i,j} \lambda_{i,j} \leq O(n)$. Although the number of constraints may be exponential, it is not hard to see that one can efficiently implement a single step of the PRA using the separation oracle. So this gives a polynomial-time algorithm. \square

By analyzing the Chernoff tail bounds, we can give one possible value for b_k to satisfy Theorem 6.1. The choice of b_k is parametrized by a choice of some small $\epsilon > 0$; this determines a multiplicative increase in b_k as compared to c_k . In addition, there is a second discrepancy term.

Theorem 6.2. *Suppose we are given an LP satisfying the requirement of Theorem 6.1, and $D \geq 1$, and $\epsilon \leq 1/(D + 1)$, and $c_k \geq 1$ for all $k \in [K]$.*

Then the following vector b_k satisfies Theorem 6.1:

$$b_k \leq \begin{cases} \frac{100 \ln(1/\epsilon)}{1 + \ln(\frac{\ln(1/\epsilon)}{c_k})} & \text{for } c_k \leq \ln(1/\epsilon) \\ c_k(1 + \epsilon) + 10 \sqrt{c_k \ln(\frac{D+1}{c_k \epsilon^2})} & \text{for } \ln(1/\epsilon) \leq c_k < \epsilon^{-2} \\ c_k(1 + \epsilon) + 10 \sqrt{c_k \ln(D + 1)} & \text{for } c_k \geq \epsilon^{-2} \end{cases}$$

Proof. Let $\delta = \ln(1/\epsilon)$ and let $d = \ln(D+1)$. As $D \geq 1$, we have $\epsilon \leq 1/2$ and hence $\delta \geq 1/2$. To simplify the notation in this proof, we omit the dependence on k and write simply b, c instead of b_k, c_k . We also write $\mu = c(1+\epsilon)$.

Case I: $c \leq \delta$. We estimate $\text{Ch}(\mu, b)$ as:

$$\begin{aligned} \text{Ch}(\mu, b) &\leq e^{b-c(1+\epsilon)} \left(\frac{c(1+\epsilon)}{b} \right)^b \leq e^b \left(\frac{2c}{b} \right)^b \\ &= \exp \left(\delta \left(-\frac{100(1 + \ln(x/50) + \ln(1 - \ln(x)))}{\ln x - 1} \right) \right) \quad \text{where } x = \delta/c \\ &\leq e^{-94.56\delta} \quad (\text{simple calculus, using the fact that } x \in [0, 1]) \end{aligned}$$

This is clearly $\leq 1/2$ for $\epsilon \leq 1/2$, so (C1) is satisfied. To show (C2):

$$\begin{aligned} \left(\frac{b+1}{c(1+\epsilon)} - 1 \right) \text{Ch}(\mu, b) \times 4D/\epsilon &\leq \left(\frac{2b}{1} \right) \text{Ch}(\mu, b) \times 4\epsilon^{-2} \\ &\leq 200\delta \text{Ch}(\mu, b) \times 4\epsilon^{-2} \\ &\leq 800\delta e^{-94.56\delta+2\delta} \\ &\leq 3.2 \times 10^{-18} \leq 1 \end{aligned}$$

Case II: $\delta < c < \epsilon^{-2}$. Let $v = (D+1)\epsilon^{-2}/c$; note $(D+1) \leq v \leq \epsilon^{-3}$. The relative deviation between μ and b here is given by $\lambda = b/\mu - 1 = 10 \frac{\sqrt{c \ln v}}{c(1+\epsilon)}$. We observe the following bound on the size of λ :

$$\lambda = 10 \frac{\sqrt{\ln v}}{\sqrt{c}(1+\epsilon)} \leq 10 \frac{\sqrt{\ln \epsilon^{-3}}}{\delta} = 10\sqrt{3} \leq 17.4$$

We can apply Proposition A.1 (which we defer to Appendix A), using $y = 17.4$, this gives us that

$$\text{Ch}(\mu, \mu(1+\lambda)) \leq e^{-\mu\lambda^2/10}$$

This gives $\text{Ch}(\mu, b) \leq v^{-\frac{10}{1+\epsilon}} \leq v^{-6.6}$. As $v \geq D+1 \geq 2$, this is ≤ 0.0103 , so (C1) is satisfied. For (C2), we have

$$\begin{aligned} \left(\frac{b+1}{c(1+\epsilon)} - 1 \right) \text{Ch}(c(1+\epsilon), b) &\leq \left(\frac{c(1+\epsilon) + 10\sqrt{c \ln v} + 1 - c(1+\epsilon)}{c(1+\epsilon)} \right) v^{-5} \\ &\leq 20 \sqrt{\frac{\ln v}{c}} v^{-6.6} \\ &\leq 10v^{-5.6} c^{-1/2} \quad \sqrt{\ln v} \leq 0.5v \text{ for } v \geq D+1 \geq 2 \\ &= 10(D+1)^{-5.6} \epsilon^{2 \times 5.6} c^{5.6-0.5} \\ &\leq 10(D+1)^{-5.6} \epsilon^{2 \times 5.6} (\epsilon^2)^{5.6-0.5} \quad \text{as } c \leq \epsilon^{-2} \\ &= \frac{\epsilon}{4(D+1)} \times 40(D+1)^{-4.6} \\ &\leq \frac{\epsilon}{4D} \times 0.255 \quad \text{as } D \geq 1 \end{aligned}$$

and so (C2) is satisfied.

Case III: $b > \epsilon^{-2}$. This is similar to Case II, but much simpler; we omit it. \square

We can simplify this expression if we assume, for instance, that the RHS vectors c have the same magnitude.

Proposition 6.3. *Suppose that we have $c_k = R$ for all k , where $R, D \geq 1$. Let $d = \ln(D + 1)$. Then we can satisfy Theorem 6.1 setting*

$$b_k \leq \begin{cases} \frac{Cd}{1 + \ln(d/R)} & R \leq d \\ R + C\sqrt{Rd} & R > d \end{cases}$$

where C is some universal constant.

Proof. Suppose that $R \leq (D + 1)^{10}$. Then apply Theorem 6.2 with $\epsilon = (D + 1)^{-10} \sqrt{d/R}$. Let $\delta = \ln(1/\epsilon)$; then $d < \delta \leq O(d)$.

Suppose $R < \delta$. Then we have

$$b_k = \frac{100\delta}{1 + \ln(\delta/c_k)} = O\left(\frac{d}{1 + \ln(d/R)}\right)$$

as desired.

If $d < R < \delta$, then $R = \Theta(\delta) = \Theta(d)$, and so we have

$$b_k = \frac{100\delta}{1 + \ln(\delta/R)} = O(d) \leq R + O(\sqrt{Rd})$$

If $\delta \geq R$, observe that $c_k \epsilon^2 = d(D + 1)^{-20} \leq 1$, and so case II applies. We have

$$b_k = R + 10\sqrt{R \log\left(\frac{D + 1}{R\epsilon^2}\right)} \leq R + 10\sqrt{R \log((D + 1)^{21})} \leq R + O(\sqrt{Rd})$$

Next we consider the case when $R > (D + 1)^{10}$. Then apply Theorem 6.2 with $\epsilon = \sqrt{d/R}$. Observe that we have $\epsilon < 1/(D + 1)$ by our condition on R . Also observe that we have $c_k \epsilon^2 = d \geq 1$ and $c_k/\delta \geq R/\ln R \geq 1$, and so case III applies giving

$$b_k = R + 10\sqrt{R \ln(D + 1)} = R + O(\sqrt{Rd})$$

□

6.2. Application to multi-dimensional scheduling. The multi-dimensional scheduling application from the introduction follows as an easy application of Theorem 6.2. First, given some candidate makespan T , we can, motivated by (3), set $z_{i,j} := 0$ if there exists some ℓ for which $p_{i,j,\ell} > T$. After this filtering, we solve the LP relaxation. If it gives a feasible solution, we scale the LP so that all the c_k values equal 1; our filtering ensures that the coefficient matrix has entries in $[0, 1]$ now, as required.

The linear constraints on the assignment are summarized as

$$\forall j \in [K], \forall \ell \in [D], \sum_i \frac{p_{i,j,\ell}}{T} z_{i,j} \leq 1;$$

For each pair (i, j) , there are D constraints, and our filtering ensures that the coefficient $\frac{p_{i,j,\ell}}{T}$ is at most 1. Hence the maximum column sum is D .

By Proposition 6.3, we can now set $b_k = O\left(\frac{\log D}{\log \log D}\right)$. This result is not new to this paper, but it is obtained in a particularly straightforward way.

6.3. Application to discrepancy. As another application, we consider a discrepancy problem introduced in [12]. Suppose we are given a $K \times n$ matrix Y , whose entries are real numbers in the range $[-1, 1]$, and which satisfies the following bounds on the ℓ_1 norms of the rows and columns:

$$\forall i \sum_k |Y_{k,i}| \leq D \quad \forall k \sum_i |Y_{k,i}| \leq R,$$

where $R, D \geq 1$.

We will show how to find a vector $v \in \{-1, +1\}^n$ such that, for all $k = 1, \dots, K$ we have

$$|Y_k \cdot v| \leq O(\sqrt{R \log(D+1)})$$

This problem was considered by [12], which gave an algorithm to construct Y via repeated quantization steps and applications of the standard LLL; this obtain the slightly weaker bound $|Y_k \cdot v| \leq O(\sqrt{R \log(R(D+1))})$. We obtain the stronger more result as a direct consequence of Proposition 6.2.

We let $d = \ln(D+1)$ and for each $k \in [K]$ let $|Y_k| = \sum_i |Y_{k,i}|$. Observe that if $|Y_k| \leq 100\sqrt{Rd}$ then it is trivial to satisfy the constraint, as for $v \in \{-1, +1\}^n$ we have $|Y_k \cdot v| \leq |Y_k| \leq 100\sqrt{Rd}$. Such constraints may then simply be ignored. So for the remainder of this proof we assume that $|Y_k| \geq 100\sqrt{Rd}$ for all k . Similarly, we assume that $R \geq 100d$ (as otherwise we necessarily have $|Y_k| < 100\sqrt{Rd}$ for all k .)

For each $i = 1, \dots, n$, we introduce a variable X_i which takes two possible values which we name $+1, -1$. Now, for each $k = 1, \dots, K$ we introduce two packing constraints

$$\begin{aligned} \sum_{i: Y_{k,i} > 0} Y_{k,i} [X_i = +1] &\leq b_k \\ \sum_{i: Y_{k,i} < 0} (-Y_{k,i}) [X_i = -1] &\leq b_k \end{aligned}$$

where we will ensure that

$$b_k \leq |Y_k|/2 + O(\sqrt{Rd})$$

Let $c_k = |Y_k|/2$ for all constraints. The LP is solvable; simply take the fractional solution defined by $z_{i,+1} = z_{i,-1} = 1/2$ for all i . Then, for any constraint k we have

$$\sum_{i: Y_{k,i} > 0} Y_{k,i} z_{i,+1} = 1/2 \sum_{i: Y_{k,i} > 0} Y_{k,i} \leq 1/2 |Y_k| = c_k$$

and similarly for $z_{i,-1}$. It is similarly easy to see that the new system has maximum ℓ_1 -column-norm of D .

Finally, if this CSP has a solution, we claim that the resulting vector v achieves the desired discrepancy. For, we have

$$\begin{aligned} Y_k \cdot v &= \sum_i Y_{k,i} ([X_i = +1] - [X_i = -1]) \\ &= \sum_{i: Y_{k,i} < 0} (Y_{k,i} - 2Y_{k,i} [X_i = -1]) + \sum_{i: Y_{k,i} > 0} (-Y_{k,i} + 2Y_{k,i} [X_i = +1]) \\ &= -|Y_k| + 2 \sum_{i: Y_{k,i} < 0} (-Y_{k,i} [X_i = -1]) + 2 \sum_{i: Y_{k,i} > 0} (Y_{k,i} [X_i = +1]) \\ &= -|Y_k| + 2 \sum_{i: Y_{k,i} < 0} (-Y_{k,i} [X_i = -1]) + 2 \sum_{i: Y_{k,i} > 0} (Y_{k,i} [X_i = +1]) \\ &\leq -|Y_k| + 2(|Y_k|/2 + O(\sqrt{Rd})) + 2(|Y_k|/2 + O(\sqrt{Rd})) \\ &\leq O(\sqrt{Rd}) \end{aligned}$$

We now apply Theorem 6.2. Suppose first that $R \leq (D+1)^{10}$. Then we take $\epsilon = (D+1)^{-10} \sqrt{d/R}$. Letting $\delta = \ln(1/\epsilon)$, we observe that $\delta \leq 15d$. As $c_k \geq |Y_k|/2 \geq 50\sqrt{Rd} \geq 50\sqrt{100d^2} \geq 500d$, we

have $c_k \geq \delta$ and so either Case II or Case III applies. In case II, we have that

$$b_k = c_k(1 + \epsilon) + O\left(\sqrt{c_k \ln\left(\frac{(D+1)}{c_k \epsilon^2}\right)}\right) \leq R + \epsilon R + O(\sqrt{R\delta}) \leq R + O(\sqrt{Rd}).$$

A similar result holds for Case III.

Next, if $R > (D+1)^{10}$, then apply Theorem 6.2 with $\epsilon = \sqrt{d/R}$; this satisfies the condition $\epsilon < 1/(D+1)$. Then, for any constraint k , we have $c_k/\delta \geq |Y_k|/2 \geq 50\sqrt{Rd}/\ln(R) \geq 1$, so case II or case III applies.

In Case III, we have $b_k \leq |Y_k|/2 + |Y_k|\epsilon + 10\sqrt{Rd} \leq R + O(\sqrt{Rd})$.

In Case II, we have $c_k \leq \epsilon^{-2}$ and

$$b_k \leq |Y_k|/2 + \epsilon|Y_k|/2 + 10\sqrt{c_k \ln\left(\frac{(D+1)}{c_k \epsilon^2}\right)}$$

Note that $\epsilon|Y_k|/2 \leq R\epsilon \leq \sqrt{Rd}$. Also it is easy to see that for $c_k \leq \epsilon^{-2}$, the expression $c_k \ln\left(\frac{(D+1)}{c_k \epsilon^2}\right)$ is an increasing function of c_k , so it can be upper bounded by its value at $c_k = \epsilon^{-2}$, namely $\epsilon^{-2} \ln(D+1)$. We thus have

$$b_k \leq |Y_k|/2 + \sqrt{Rd} + 10\sqrt{\epsilon^{-2} \ln(D+1)} = |Y_k|/2 + \sqrt{Rd} + 10\sqrt{R} \leq |Y_k|/2 + O(\sqrt{Rd})$$

as desired.

6.4. Comparison with the MT algorithm. To compare our results with the MT algorithm, let us consider the class of assignment-packing problems in which we have $c_k = R$ for all k . We have seen (Proposition 6.3) that the PRA converges in polynomial time if the bad-events are defined with RHS vector $b_k = R'$; here R' is a function of R and of the maximum ℓ_1 -norm D of the constraint matrix. Crucially, R' is scale-free: it does *not* depend on the number of variables n or number of constraints m .

In this section, we show that MT cannot be used to achieve *any* value b_k which depends solely on R, D . In fact, we show that MT cannot achieve even a value of b_k parametrized in terms of D' (which may be much larger than D). Thus, the MT algorithm is qualitatively much weaker than the PRA in this case.

We note that there are many possible parametrizations of the LLL and the MT algorithm for this problem, including strategies based on iterated applications. We will only consider the simplest parametrization, in which there is a separate bad-event for each row of the constraint matrix. Also, we note that our analysis of the PRA algorithm guarantees the existence of a solution, and the MT algorithm will (at least heuristically) eventually find it after exploring the full solution space. Thus, we show that the MT algorithm requires exponential time to converge; we cannot hope to show that it never converges.

More formally, we show the following:

Proposition 6.4. *Let R, R' be fixed real numbers with $1 \leq R \leq R'$. For m sufficiently large there is an assignment-packing problem with the following properties:*

- (1) *The system has n variables and m constraints, where $n = \Theta(Rm)$.*
- (2) *There is a fractional solution z achieving RHS value $c_k = R$.*
- (3) *For each $i \in [n]$ and each possible assignment j to that variable, there is exactly one row k with $a_{k,i,j} > 0$. (So the matrix has maximum ℓ_0 norm of $D' = 1$.)*
- (4) *Suppose we run the PRA, using resampling probabilities given by $p_{ij} = z_{ij}$, and where the targeted bad-events are defined using RHS value $b_k = R + C\sqrt{R}$, where C is a universal constant. Then the PRA algorithm (with the appropriate choice of fractional hitting-set) terminates in expected polynomial time.*

- (5) Suppose we run the MT algorithm, using resampling probabilities given by $p_{ij} = z_{ij}$ and where the targeted bad-events are defined using RHS value $b_k = R'$. Then the probability that this MT algorithm terminates after $2^{\phi m}$ steps is at most $2^{-\phi' m}$; here $\phi, \phi' > 0$ are parameters which depend solely on R, R' .

Proof. There are n variables, with each variable $i \in [n]$ taking values in the range $L_i = \{1, \dots, m\}$. To construct the constraint matrix A , we select for each $i \in [n]$ a permutation $\pi_i \in S_m$ independently and uniformly at random. For $k = 1, \dots, m$ we then set

$$a_{k,i,j} = [\pi_i(k) = j]$$

Thus, all the entries of the constraint matrix are either 0 or 1. Also, observe that for any i, j , the only value of k such that $a_{k,i,j} > 0$ is given by $k = \pi_i^{-1}(j)$.

We claim now that this system has an LP solution; set $z_{i,j} = 1/m$ for all i, j . Then $\sum_j z_{i,j} = 1$ and for any k we have

$$\sum_{i,j} a_{k,i,j} z_{i,j} = \sum_{i,j} [\pi_i(k) = j]/m = n/m$$

For $n = \lfloor Rm \rfloor$, this satisfies the constraints fractionally with RHS vector $c_k = R$.

Furthermore, the convergence of the PRA with this fractional vector z , follows from Proposition 6.3.

Finally, we will show that the MT algorithm requires a long time to terminate. To show this, we claim the following: suppose $x_1, \dots, x_n \in [m]^n$ is a fixed vector. Then the probability (over the random choice of A) that $A_k x \leq R'$ for all $k = 1, \dots, m$, is at most $e^{-\Omega(m)}$.

To show this, note that we can view the vector of counts $A_1 x, \dots, A_m x$ as what is known as a *competing ball-and-urns problem*; there is an urn corresponding to each $k = 1, \dots, m$, there is a ball corresponding to each $i = 1, \dots, n$, and the value of $A_k x$ is the number of balls placed into urn k . We place ball i into urn k iff $\pi_i(k) = x_i$ — in other words, the placement of each ball is independently chosen among the k urns.

Consider some fixed value of k . The value of $A_k x$ is a binomial random variable, with number of trials n and success probability $1/m$. Thus, the probability that this exceeds R' is given by

$$\begin{aligned} P(A_k x > R') &= \sum_{\ell=R'+1}^n \binom{n}{\ell} (1/m)^\ell (1-1/m)^{n-\ell} \\ &\geq \binom{n}{R'+1} (1/m)^{R'+1} (1-1/m)^{n-R'-1} \\ &\geq \frac{(n-R')^{R'+1} (1-1/m)^{n-R'-1}}{(R'+1)! m^{R'+1}} \\ &\geq \frac{(Rm-1-R')^{R'+1} (1-1/m)^{Rm-R'-1}}{(R'+1)! m^{R'+1}} \quad \text{as } Rm-1 \leq n \leq Rm \end{aligned}$$

Now, simple calculus show that for fixed $1 \leq R \leq R'$, this approaches to a limit of $\frac{e^{-R} R^{R'+1}}{(R'+1)!}$ as $m \rightarrow \infty$. Thus, for R, R' fixed and m sufficiently large, we have that $P(A_k x > R') \geq \Omega(1)$.

Furthermore, as shown in [7], the events $A_1 x \leq R', \dots, A_m x \leq R'$ are negatively correlated. Thus, we have that

$$P\left(\bigwedge_{k=1}^m A_k \leq R'\right) \leq \prod_{k=1}^m P(A_k \leq R') \leq \prod_{k=1}^m \left(1 - (1 - \Omega(1))\right) \leq e^{-\Omega(m)}$$

Now, we observe that every row in the CSP depends on every variable. Thus, whenever the MT resamples a bad-event, it completely resamples every variable. As a result of this, after the MT

performs T resamplings, then the current value of the variables X_1, \dots, X_n is simply the T^{th} row of the resampling table.

As a consequence of this, a necessary condition for the MT algorithm to terminate after T steps is that one of the first T rows of the resampling table satisfies all the constraints. The probability that any individual row satisfies all the constraints is $\leq e^{-\Omega(m)}$, so by the union-bound the probability that one of the first T rows satisfies the constraints is $\leq Te^{-\Omega(m)}$. This is at most $e^{-\Omega(m)}$ unless $T \geq e^{\Omega(m)}$. \square

7. PACKET ROUTING

We begin by reviewing the basic strategy of [20], and its improvements by [28] and [25]. We recommend consulting [28], which is a very readable presentation of this problem as well as many more details and variants than we cover here. We note that [25] studied a more general version of the packet-routing problem, so their choice of parameters was not (and could not be) optimized.

We are given a graph G with N packets. Each packet has a simple path, of length at most D , to reach its endpoint vertex (we refer to D as the *dilation*). In any timestep, a packet may wait at its current position, or move along the next edge on its path. Our goal is to find a schedule of smallest makespan in which, in any given timestep, an edge carries at most a single packet.

We define the *congestion* C to be the maximum, over all edges, of the number of packets scheduled to traverse that edge. It is clear that D and C are both lower bounds for the makespan, and [20] has shown that in fact a schedule of makespan $O(C + D)$ is possible. [28] provided an explicit constant bound of $39(C + D)$, as well as describing an algorithm to find such a schedule. This was improved to $23.4(C + D)$ in [25] as will be described below.

While the final schedule only allows one packet to cross an edge at a time, we will relax this constraint during our construction. We consider “infeasible” schedules, in which arbitrarily many packets pass through each edge at each timestep. We define an *interval* to be a consecutive set of times in our schedule, and the *congestion* of an edge in a given interval to be the number of packets crossing that edge. If we are referring to intervals of length i , then we define a *frame* to be an interval which starts at an integer multiple of i .

From our original graph, one can easily form an (infeasible) schedule with delay D and overall congestion C . Initially, this congestion may “bunch up” in time, that is, certain edges may have very high congestion in some timesteps and very low congestion in others. So the congestion is not bounded on any smaller interval than the trivial interval of length D . During our construction, we will “even out” the schedule, bounding the congestion on successively smaller intervals.

Ideally, one would eventually finish by showing that on each individual timestep (i.e. interval of length 1), the congestion is roughly C/D . In this case, one could turn such an infeasible schedule into a feasible schedule, by simply expanding each timestep into C/D separate timesteps.

As [25] showed, it suffices to control the congestion on intervals of length 2. Given our infeasible schedule, we can view each interval of length 2 as defining a new subproblem. In this subproblem, our packets start at a given vertex and have paths of length 2. The congestion of this subproblem is exactly the congestion of the schedule. Hence, if we can schedule problems of length 2, then we can also schedule the 2-intervals of our expanded schedule.

We quote the following result from [25].

Proposition 7.1 ([25]). *Suppose there is an instance with delay $D = 2$ and congestion C . Then there is a schedule of makespan $C + 1$, which can be found in polynomial time.*

This was used by [25] to improve the bound on the overall makespan to $23.4(C + D)$. [25] speculated that by examining the scheduling for longer, but still small, delays, one could further improve the general packet routing. Unfortunately, we are not able to show a general result for small delays such as $D = 3$. However, as we will see, the schedules that are produced in the larger

construction of [28] are far from generic, but instead have relatively balanced congestion across time. We will see how to take advantage of this balanced structure to improve the scheduling.

7.1. Using the LLL to find a schedule. The general strategy for our construction is based on [28] with some small changes to parameters. We will add random delays to each packet, and then allow the packet to move through each of its edges in turn without hesitation. This effectively homogenizes the congestion across time. We use the LLL to ensure that the congestion does not get too large on any interval.

Lemma 7.2. *Let $1 \leq i' < i$, and let m, C' be non-negative integers. Suppose there is a schedule S of length L such that every interval of length i has congestion at most C . Suppose that we have*

$$e \times P(\text{Binomial}(C, \frac{i'}{i-i'}) > C') \times (Cmi^2 + 1) < 1$$

Then there is a schedule S' of length $L' = L(1 + 1/m) + i$, in which every interval of length i' has congestion $\leq C'$. Furthermore, this schedule can be constructed in expected polynomial time.

Proof. We break the schedule S into frames of length $F = mi$, and refine each separately. Within each frame, we add a random delay to each packet separately. The delays are uniformly distributed in the range $\{0, \dots, i - i'\}$ and are independent. (We refer to this as *adding a random delay in the range $i - i'$ to each packet*)

Let us fix an F -frame for the moment. Associate a bad event to each edge f and i' -interval I , that the congestion in that interval and edge exceeds C' .

For each I, f , there are at most C possible packets that could cross, and each does so with probability $p = \frac{i'}{i-i'}$. Hence the probability of the bad event is at most the probability that a Binomial random variable with C trials and probability p exceeds C' .

Next consider the dependency. Given an edge f and i' -interval I , there are at most C packets crossing it, each of which may pass through up to mi other edges in the frame. We refer to the combination of a specific packet passing through a specific edge as a *transit*. Now, for each transit, there are (depending on the delay assigned to that packet) at most i other i' -intervals in which this transit could have been scheduled. Hence the dependency is at most Cmi^2 .

By the LLL, the condition in the hypothesis guarantees that there is a positive probability that the delays avoid all bad events. In this case, we refine each frame of S to obtain a new schedule S' as desired. We can use the algorithmic LLL to actually find such schedules S' in polynomial time.

So far, this ensures that *within each frame*, the congestion within any interval of length i' is at most C' . In the refined schedule S' there may be intervals that cross frames. To ensure that these do not pose any problems, we insert a delay of length i' between successive frames, during which no packets move at all. This step means that the schedule S' may have length up to $L(1 + 1/m) + i$. \square

Using this Lemma 7.2, we can transform the original problem instance (in which C, D may be unbounded), into one in which C, D are small finite values. In order to carry out this analysis properly, one would need to develop a series of separate bounds depending on the sizes of C, D . To simplify the exposition, we will assume that C, D are very large, in which case certain rounding effects can be disregarded. When C, D are smaller, we can show stronger bounds but doing this completely requires extensive case analysis of the parameters.

Lemma 7.3. *Assume $C + D \geq 2^{896}$. There is a schedule of length at most $1.004(C + D)$ and in which the congestion on any interval of length 2^{24} is at most 17040600. Furthermore, this schedule can be produced in polynomial time.*

Proof. Define the sequence a_k recursively as follows.

$$a_0 = 256 \quad a_{k+1} = 2^{a_k}$$

There is a unique k such that $a_k^{3.5} \leq (C + D) < a_{k+1}^{3.5}$. By a slight variant on Lemma 7.2, one can add delays to obtain a schedule of length $C + D$, in which the congestion on any interval of length $i' = a_k^3$ is at most $C' = i'(1 + 4/a_k)$.

At this point, we use Lemma 7.2 repeatedly to ensure to control the congestion intervals of length a_j^3 , for $j = k - 1, \dots, 0$. At each step, this increases the length of the resulting schedule from L_j to $L_j(1 + 1/a_{j+1}) + a_j$, and increases the congestion on the relevant interval from $i(1 + 4/a_k)$ to

$$i(1 + 4/a_k) \prod_{j=0}^{k-1} (1 + 4/a_j) \left(\frac{1}{1 - (a_j/a_{j+1})^3} \right)$$

(We use the Chernoff bound to estimate the binomial tail in Lemma 7.2.)

For $C + D \geq a_k^{3.5}$, it is a simple calculation to see that the increase in length is from $C + D$ (after the original refinement) to at most $1.004(C + D)$. In the final step of this analysis, we are bounding the congestion of intervals of length $a_0^3 = 2^{24}$, and the congestion on such an interval is at most 17040600.

Furthermore, since all of these steps use the LLL, one can form all such schedules in polynomial time.

See [28] for a much more thorough explanation of this process. \square

Now that we have reduced to constant-sized intervals, we are no longer interested in asymptotic arguments, and come down to specific numbers.

Lemma 7.4. *There is a feasible schedule of length at most $10.92(C + D)$, which can be constructed in polynomial time.*

Proof. For simplicity, we assume $C + D \geq 2^{896}$.

By Lemma 7.3, we form a schedule S_1 , of length $L_1 \leq 1.004(C + D)$, in which each interval of length 2^{24} has congestion at most 17040600.

Now apply Lemma 7.2 to S_1 , with $m = 64, i' = 1024, C' = 1385$ to obtain a schedule S_2 , of length $L_2 \leq 1.0157L_1 + 2^{24}$, in which each interval of length 1024 has congestion at most 1385.

Now apply Lemma 7.2 to S_2 with $m = 64, i' = 2, C' = 20$, to obtain a schedule S_3 of length $L_3 \leq 1.0157L_2 + 1024$, in which each frame of length 2 has congestion at most 20.

Now apply Proposition 7.1 to S_3 , expanding each 2-frame to a *feasible* schedule of length 21. The total length of the resulting schedule is at most $\frac{21}{2}L_3 \leq 10.92(C + D)$. \square

7.2. The PRA applied to packet routing. So far, all of the improvements we have made to the packet routing problem used nothing more than the conventional LLL. We now show how to modify this construction to use the PRA in the appropriate places.

Let us examine more closely the process used to refine a schedule in which each interval of length C has congestion at most i . We break the schedule S into frames of length $F = mi$, and refine each separately. Within each frame, we add a random delay of range $b = i - i'$ to each packet separately. Each F -frame gives us what is essentially an assignment-packing problem: we give set a delay to each packet, and a bad events correspond to an edge receiving an excessive congestion in some time interval.

We can modify Lemma 7.3 and 7.4 to use Theorem 3.3 instead of the LLL.

Proposition 7.5. *Let $i' < i$, let m, C', d be non-negative integers with $d \leq C'$. Suppose there is a schedule S of length L such that every interval of length i has congestion at most C .*

Define

$$p = \frac{(Ci'\alpha)^d}{d! \binom{C'+1}{d}}$$

Suppose we have $p < 1$ and

$$(i - i')\alpha - \frac{mi^2dp}{C(1-p)} \geq 1$$

Then there is a schedule S' of length $L' = L(1 + 1/m) + i$, in which every interval of length i' has congestion $\leq C'$. Furthermore, such a schedule can be found in polynomial time.

Proof. Suppose we add delays in the range $b = i - i' - 1$ uniformly to each packet within each frame of length $F = mi$. In this case, the categories correspond to each packet x , and for each delay t we assign $\lambda_{x,t} = \alpha$. For each edge f and i' -interval I , we introduce a complex bad event $\mathcal{B}_{f,I}$ that the congestion in the interval exceeds C' . Each such bad-event has a distinct label, and we use the fractional hitting-set $Q_{f,I}$ of width d as described in Theorem 4.3.

For a given f, I , we must compute the total contribution of λ summed over all packets/delays which could contribute to the congestion of that edge-interval. There are at most C packets which could be scheduled to pass through the given edge, and there are i' possible delays which affect f, I . So, in all, the total contribution is $\mu \leq Ci'\alpha$. The bad event is that this exceeds C' , so we have here $t = C' + 1$. By Theorem 4.3, this gives $G(Q_{f,I}, \lambda) \leq \frac{\mu^d}{d!(C'+1)^d} = p$

Next, consider some packet x . This can affect up to mi edges in the F -frame, each of which affects at most i intervals of length i' . For each resulting f, I affected by x , we have $\mu_i \leq i'\alpha$ and hence $G_x(Q_{f,I}, \lambda) \leq \frac{i'\alpha}{\mu} d \frac{\mu^d}{d!(C'+1)^d} \leq \frac{dp}{C}$. Hence, summing $G_x(Q_{f,I}, \lambda)$ over all f, I affected by packet x , we have

$$G_x(Q_{f,I}, \lambda) \leq \frac{mi^2dp}{C}$$

By Theorem 3.3(b), in order for the PRA to produce a satisfying solution, we must satisfy for each packet x the constraint

$$(14) \quad \lambda_x \geq 1 + \sum_k \frac{G_i(Q_k, \lambda)}{1 - G(Q_k, \lambda)}.$$

We have $\lambda_x = (i - i')\alpha$ and $G(Q_k, \lambda) \leq p$ and $\sum_k G_i(Q_k, \lambda) \leq mi^2dp/C$, so (14) becomes

$$(i - i')\alpha \geq 1 + \frac{mi^2dp}{(1-p)C}$$

This is precisely the constraint specified in the hypothesis. \square

We can use this to improve various steps in the construction.

Proposition 7.6. *Suppose $C + D \geq 2^{896}$. Then there is a schedule of length $\leq 1.0157(C + D)$, in which every interval of length 1024 has congestion at most 1312, which can be constructed in polynomial time.*

Proof. By Lemma 7.3, we form a schedule S_1 , of length $L_1 \leq 1.004(C + D)$, in which each interval of length 2^{24} has congestion at most 17040600. Apply Proposition 7.5 with $\alpha = 5.985 \times 10^{-8}$, $C' = 1312$, $d = 247$, $m = 64$ to obtain a schedule S_2 of length $L_2 \leq 1.0157L_1 + 2^{24}$, in which each interval of length 1024 has congestion at most 1312. \square

Theorem 7.7. *Suppose $C + D \geq 2^{896}$. Then there is a schedule of length at most $8.77(C + D)$ which can be constructed in polynomial time.*

Proof. By Proposition 7.6, there is a schedule S_1 of length at most $1.0157(C + D)$ in which each interval of length 1024 has congestion at most 1312.

Now apply Lemma 7.3 with $i = 1024$, $C = 1312$, $i' = 2$, $m = 64$, $C' = 16$, $d = 12$, $\alpha = 0.0011306$ to obtain a schedule S_2 of length $L_2 \leq 1.0157L_1 + 1024$, in which each interval of length 2 has congestion at most 16.

Now apply Proposition 7.1 to S_2 , expanding each 2-frame to a *feasible* schedule of length 17. The total length of the resulting schedule is at most $\frac{17}{2}L_2 \leq 8.77(C + D)$. \square

7.3. Better scheduling of the final 2-frame. Let us examine the last stage in the construction more closely. In this phase, we are dividing the schedule into intervals of length 2, and we want to control the congestion of each edge in each 2-frame.

For a given edge f and time t , we let $c_t(f)$ denote the number of packets scheduled to cross that edge in the four time steps of the original (infeasible) schedule.

Suppose we have two consecutive 2-frames starting at time t . The reason for the high value of C' in the final step of the above construction is that it is quite likely that $c_t + c_{t+1}$ or $c_{t+2} + c_{t+3}$ are much larger than their mean. However, it would be quite rare for *both* these bad events to happen simultaneously. We will construct a schedule in which we insert an “overflow” time between the 2-frames. This overflow handles cases in which either $c_t + c_{t+1}$ is too large *or* $c_{t+2} + c_{t+3}$ is too large.

Our goal will be to modify either of the 2-frames so as to ensure that the congestion is at most T . In order to describe our modification strategy, we first fix, for every packet and frame, a “first edge” and “second edge” in this frame. Some packets may only transit a single edge, which we will arbitrarily label as first or second. As we modify the schedule, some packets that initially had two transits scheduled will be left with only one; in this case, we retain the label for that edge. So, we may assume that every edge is marked as first or second and this label does not change.

We do this by shifting transits into the overflow time. For each 2-frame, there are two overflow times, respectively earlier and later. If we want to shift an edge to the later overflow time, we choose any packet that uses that edge as a second edge (if any), and reschedule the second transit to the later overflow time; similarly if we shift an edge to the earlier overflow time. See Figure 1.

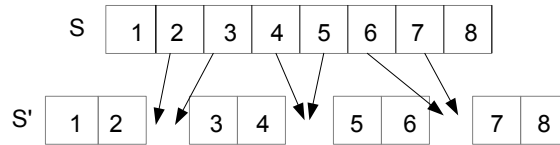


FIGURE 1. The packets in the original schedule S are shifted into overflow times in the schedule S' .

Note that in the analysis of [25], the only thing that matters is the *total* congestion of an edge in each 2-frame. In deciding how to shift packets into the overflow times, we need to be careful to account for how often the edge appears as the first or second transit. If an edge appears exclusively as a “first edge”, we will only be able to shift it into the earlier overflow, and similarly if an edge appears exclusively as a “second edge”.

Keeping this constraint in mind, our goal is to equalize as far as possible the distribution of edges into earlier and later overflows. We do this by the following scheme:

1. For each edge f and every odd integer $t = 1, 3, 5, \dots, L$, repeat while $c_t(f) + c_{t+1}(f) > T$:
 2. If $c_t(f) = 0, c_{t+1}(f) > T$, then shift one packet into the later overflow time.
 3. Else if $c_t(f) > T, c_{t+1}(f) = 0$, then shift one packet into the earlier overflow time.
 4. Else if $c_t(f) + c_{t+1}(f) > T, c_t(f) > 0, c_{t+1}(f) > 0, c_t(f) + c_{t+1}(f) = \text{odd}$, then shift one packet into the earlier overflow time.
 5. Else if $c_t(f) + c_{t+1}(f) > T, c_t(f) > 0, c_{t+1}(f) > 0, c_t(f) + c_{t+1}(f) = \text{even}$, then shift one packet into the later overflow time.

Suppose we fix t to be some odd integer. If we let c' denote the congestions at the end of this overflow-shifting process, then we have $c'_t(f) + c'_{t+1}(f) \leq T$, and the number of packets shifted into

the earlier (respectively later) overflow time can be viewed as a function of the original values of the congestions c_t, c_{t+1} . We denote these “overflow” functions by $OF^-(c_t, c_{t+1}; T)$ and $OF^+(c_t, c_{t+1}; T)$ respectively.

Specifically we get the following condition:

Proposition 7.8. *Suppose that we have a schedule of even length L , and let $c_t(f)$ for $t = 1, \dots, L$ denote the number of times f is scheduled as the t^{th} edge of a packet. Suppose that for all edges $f \in E$ and all $t = 1, 3, 5, \dots$ we satisfy the constraint*

$$OF^+(c_t(f), c_{t+1}(f); T) + OF^-(c_{t+2}(f), c_{t+3}(f); T) \leq T'$$

as well as the boundary constraints

$$OF^-(c_1(f), c_2(f); T) \leq T' \quad OF^+(c_{L-1}(f), c_L(f); T) \leq T'$$

Then there is a schedule of makespan $L \times \frac{T+T'+2}{2} + T'$, which can be constructed in polynomial time.

Proof. After the modification, each 2-frame has congestion at most T , while each overflow time has congestion at T' . Each overflow time has delay at most 2, since for any packet x , there may be at most two edges scheduled into that overflow time, namely the edge that had been originally marked as the second edge of the earlier 2-frame, and the edge that had been originally marked as the first edge of the latter 2-frame. Hence each 2-frame can be scheduled in time $T + 1$ and each overflow can be scheduled in time $T' + 1$. As there are $L/2$ 2-frames in the original schedule, there are $L/2 + 1$ overflow periods. Hence the total cost is at most $L \frac{T+T'+2}{2} + T'$. \square

Note that the conditions required by this Proposition 7.8 are local, in the sense that any violation is any event which affects an individual edge and a 4-interval which starts at an odd time t . We refer to such an interval for simplicity as an *aligned 4-interval*. We refer to the conditions required by this Proposition as the *4-conditions*; these conditions can be viewed as either pertaining to an entire schedule, or to an individual aligned 4-interval. We also note that the 4-conditions are *monotone*, in the sense that if a configuration violates them, then it will continue to do so if the congestion of any edge at any time is increased.

We will show how to use the PRA to find a schedule satisfying the conditions of Proposition 7.8.

Proposition 7.9. *Let $m = 39, C = 1312, i = 1024, T = 6, T' = 5$.*

Suppose there is a schedule S of length L such that every interval of length i has congestion at most C . There is a schedule of length $L' \leq (1 + 1/m)L + i$, which satisfies the 4-conditions with respect to T, T' . This schedule can be produced in polynomial time.

Proof. Our plan is to break the schedule into frames of size $F = mi$; within each packet and frame we add a random delay in the range $i - 4$. Let us fix a frame for the moment.

For each edge f , and each aligned 4-interval I starting at time s , we introduce a complex bad event $\mathcal{B}_{f,I}$ that

$$OF^+(c_s(f), c_{s+1}(f); T) + OF^-(c_{s+2}(f), c_{s+3}(f); T) > T'$$

For this edge/interval f, I , and any packet x with delay t , we say that $\langle x, t \rangle$ has *type j* , if that packet-delay assignment would cause the given packet x to land at position $s + j$ within the bad event, for $j = 0, \dots, 3$. If that assignment x, s does not contribute to $\mathcal{B}_{f,I}$, then $\langle x, s \rangle$ has no type. Note that for each f, I there are at most C packet-delay combinations of each type.

For a bad event $\mathcal{B}_{f,I}$ and a fractional hitting-set $Q_{f,I}$, we define the quantity $\Phi_j(f, I)$, for $j = 0, 1, 2, 3$, as

$$\Phi_j(f, I) = \max_{\substack{\langle x, t \rangle \text{ has type } j \\ \text{for } f, I}} \sum_{Y \ni \langle x, t \rangle} Q_{f,I}(Y) \lambda^Y$$

Similarly, we define $\Phi(f, I) = \sum_Y Q_{f,I}(Y) \lambda^Y$. We likewise define $\Phi_j = \max_{f,I} \Phi_j(f, I)$ and $\Phi = \max_{f,I} \Phi(f, I)$.

We will apply the PRA using a separate label for each f, I , and a separate variable for each packet (the value of a variable is the chosen delay), and the vector $\vec{\lambda}_{x,t} = \alpha = 0.001051$. For each such label $k = (f, I)$ we have $G(Q_{f,I}, \lambda) \leq \sum_Y Q_{f,I}(Y) \lambda^Y \leq \Phi$.

For any packet x and delay t and $j = 0, \dots, 3$, there are at most $mi/2$ pairs f, I for which packet x, t has type j . For each such f, I , we have $G_{x,t}(Q_{f,I}, \lambda) \leq \Phi_j$. As there are i choices for the delay t , we have that summing over all t and all such f, I gives

$$\sum_{f,I} G_x(Q_{f,I}, \lambda) \leq \frac{mi^2}{2} (\Phi_0 + \Phi_1 + \Phi_2 + \Phi_3)$$

By Theorem 3.3(b), we must satisfy the condition for each packet x

$$\lambda_x \geq 1 + \sum_{f,I} \frac{G_x(Q_{f,I}, \lambda)}{1 - G(Q_{f,I}, \lambda)}$$

We have $\lambda_x = (i - 4)\alpha$, and so it suffices to satisfy the condition

$$(15) \quad (i - 4)\alpha - \frac{mi^2}{2} \frac{\Phi_0 + \Phi_1 + \Phi_2 + \Phi_3}{1 - \Phi} \geq 1$$

in order to find acceptable delays. Such delays lead to a schedule of length $L' \leq (1 + 1/m)L + i$, which satisfies the 4-conditions with T, T' .

Thus, we have reduced our problem to constructing fractional hitting sets $Q_{f,I}$ which have a sufficiently small value for $\frac{\Phi_0 + \Phi_1 + \Phi_2 + \Phi_3}{1 - \Phi}$. We will describe this next. Although we have stated the proposition for a particular choice of parameters, we will walk through the algorithm we use to construct it.

The bad event depends $\mathcal{B}_{f,I}$ is determined by the number of variables of each type assigned to edge f on the aligned 4-interval I . There are at most C such variables of each type; to simplify the notation, we suppose there are exactly C . The fractional hitting-set $Q_{f,I}$ assigns weights to any subset of the $4C$ variables involved in the bad event; we can write such a subset as $Y = Y_1 \cup Y_2 \cup Y_3 \cup Y_4$, where the packet/delays in Y_j all have type j .

We will make $Q_{f,I}$ symmetric, in the sense that for any such $Y = Y_0 \cup Y_1 \cup Y_2 \cup Y_3$, the $Q_{f,I}(Y)$ depends solely on the cardinalities $|Y_0|, |Y_1|, |Y_2|, |Y_3|$. Thus, we define

$$Q_{f,I}(Y_0 \cup Y_1 \cup Y_2 \cup Y_3) = b(|Y_0|, |Y_1|, |Y_2|, |Y_3|)$$

where $b : [C]^4 \rightarrow [0, 1]$ is a function which we will determine. We may then compute

$$\Phi_0 \leq \sum_{y_0, y_1, y_2, y_3} \binom{C-1}{y_0-1} \binom{C}{y_1} \binom{C}{y_2} \binom{C}{y_3} b(y_0, y_1, y_2, y_3) \alpha^{y_0+y_1+y_2+y_3}$$

and similarly for $\Phi_1, \Phi_2, \Phi_3, \Phi$; here y_0, y_1, y_2, y_3 denote the possible cardinalities of Y_0, Y_1, Y_2, Y_3 respectively.⁴

In order to satisfy the PRA criterion, the fractional hitting-set must satisfy $\sum_{Y \subseteq A} Q(Y) \geq 1$ for any atomic bad event $A \in \mathcal{B}_{f,I}$. By symmetry, this means that if we have k_0, k_1, k_2, k_3 minimal such that

$$\text{OF}^+(k_0, k_1; T) + \text{OF}^-(k_2, k_3; T) > T',$$

⁴The reason for the term $\binom{C-1}{y_0-1}$ here, as opposed to $\binom{C}{y_0}$, is that in computing Φ_0 , we have fixed the presence of a single packet/delay $\langle x, t \rangle$ with type 0 for the given f, I . Thus, there are only $\binom{C-1}{y_0-1}$ choices for the *additional* type-0 packets involved in f, I .

then we require

$$(16) \quad \sum_{y_0, y_1, y_2, y_3} \binom{k_0}{y_0} \binom{k_1}{y_1} \binom{k_2}{y_2} \binom{k_3}{y_3} b(y_0, y_1, y_2, y_3) \geq 1$$

We are trying to satisfy $(\Phi_0 + \Phi_1 + \Phi_2 + \Phi_3)/(1 - \Phi) \leq t$, where t is some target value. For a fixed value of t , this is equivalently to minimizing $\Phi_0 + \Phi_1 + \Phi_2 + \Phi_3 + t\Phi$. If we view the collection of all values $b(y_0, y_1, y_2, y_3)$ as linear unknowns, then we can view both the objective function and the constraints as linear. Hence this defines a linear program, which we can solve using standard linear programming algorithms. Given that we can optimize the resulting linear program for any fixed value of t , we can also optimize over t using standard numerical algorithms.

For any y_0, y_1, y_2, y_3 , we will set $b(y_0, y_1, y_2, y_3) = 0$ unless there is some such minimal bad $k_0, k_1, k_2, k_3 \geq y_0, y_1, y_2, y_3$. This greatly reduces the number of variables we need to consider, to something which is very large but tractable. For $T = 6, T' = 5$, for instance, the linear program has 12000 variables and 259 constraints. This is too large to write explicitly, but we wrote computer code which generates and solves this LP. The resulting hitting-set achieves a bound of

$$\frac{\Phi_0 + \Phi_1 + \Phi_2 + \Phi_3}{1 - \Phi} \leq 3.495 \times 10^{-9}$$

and this satisfies (15). (It is too large to enumerate the solution explicitly; we recommend that any reader who wishes to recover it should construct the linear program of (16) and solve it for themselves.) \square

We now apply this construction to replace the two final steps in the construction of Section 7.3.

Theorem 7.10. *There is a feasible schedule of makespan at most $6.78(C + D)$, which can be constructed in expected polynomial time.*

Proof. For simplicity, we assume $C + D \geq 2^{896}$. By Proposition 7.6, we obtain a schedule S_1 of length $L_1 \leq 1.0157(C + D)$, in which each interval of length 1024 has congestion at most 1312.

Apply Proposition 7.9. This gives a schedule S_2 of length $L_2 \leq 1.0158L_1 + 1024$ satisfying the 4-conditions with $T = 6, T' = 5$. By Proposition 7.8, this yields a schedule whose makespan is $6.67L_2 + 5 \leq 6.78(C + D)$. \square

8. ACKNOWLEDGMENTS

We thank Tom Leighton and Satish Rao for valuable discussions long ago, which served as the foundation for this work; but for their declining, they would be coauthors of this paper. We are thankful to Noga Alon, Venkatesan Guruswami, Bernhard Haeupler, Penny Haxell, and Jeff Kahn for helpful discussions, as well as to the STOC 2013, FOCS 2013, and journal referees for their valuable comments.

Aravind Srinivasan dedicates this work to the memory of his late grandmother Mrs. V. Chellam-mal (a.k.a. Rajalakshmi): memories of her remain a great inspiration.

REFERENCES

- [1] R. Aharoni, E. Berger, and R. Ziv. Independent systems of representatives in weighted graphs. *Combinatorica*, 27:253–267, 2007.
- [2] N. Alon. The linear arboricity of graphs. *Israel Journal of Mathematics*, 62:311–325, 1988.
- [3] N. Alon. The strong chromatic number of a graph. *Random Structures and Algorithms*, 3:1–7, 1992.
- [4] Y. Azar and A. Epstein. Convex programming for scheduling unrelated parallel machines. In *STOC '05: Proceedings of the 36th annual ACM Symposium on Theory of Computing*, pages 331–337. ACM, 2005.
- [5] R. Bissacot, R. Fernandez, A. Procacci, and B. Scoppola. An improvement of the Lovász Local Lemma via cluster expansion, 2011.
- [6] B. Bollobás, P. Erdős, and E. Szemerédi. On complete subgraphs of r -chromatic graphs. *Discrete Math.*, 1:97–107, 1975.

- [7] D. Dubhashi and D. Ranjan. Balls and bins: a study in negative dependence. *BRICS Report Series*, 3-25, 1996.
- [8] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and Finite Sets*, volume 11 of *Colloq. Math. Soc. J. Bolyai*, pages 609–627. North-Holland, 1975.
- [9] B. Haeupler, B. Saha, and A. Srinivasan. New Constructive Aspects of the Lovász Local Lemma. *Journal of the ACM*, 58, 2011.
- [10] D. G. Harris and A. Srinivasan. Constraint satisfaction, packet routing, and the Lovász Local Lemma. In *Proc. ACM Symposium on Theory of Computing*, pages 685–694, 2013.
- [11] D. G. Harris and A. Srinivasan. The Moser-Tardos Framework with Partial Resampling. In *FOCS*, pages 469–478, 2013.
- [12] N. Harvey. A note on the discrepancy of matrices with bounded row and column sums. <http://arxiv.org/abs/1307.2159>, 2013.
- [13] P. Haxell and T. Szabó. Odd independent transversals are odd. *Comb. Probab. Comput.*, 15(1-2):193–211, January 2006.
- [14] P. E. Haxell. A note on vertex list colouring. *Combinatorics, Probability, and Computing*, 10:345–348, 2001.
- [15] P. E. Haxell, T. Szabó, and G. Tardos. Bounded size components – partitions and transversals. *Journal of Combinatorial Theory, Series B*, 88:281–297, 2003.
- [16] G. Jin. Complete subgraphs of r -partite graphs. *Combin. Probab. Comput.*, 1:241–250, 1992.
- [17] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, 2:113–129, 1987.
- [18] K. Kolipaka and M. Szegedy. Moser and Tardos meet Lovász. In *Proceedings of ACM STOC*, pages 235–244, 2011.
- [19] F. T. Leighton, C.-J. Lu, S. B. Rao, and A. Srinivasan. New Algorithmic Aspects of the Local Lemma with Applications to Routing and Partitioning. *SIAM Journal on Computing*, 31:626–641, 2001.
- [20] F. T. Leighton, B. M. Maggs, and S. B. Rao. Packet routing and jobshop scheduling in $O(\text{congestion} + \text{dilation})$ steps. *Combinatorica*, 14:167–186, 1994.
- [21] J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
- [22] P.-S. Loh and B. Sudakov. Independent transversals in locally sparse graphs. *Journal of Combinatorial Theory, Series B*, 97:904–918, 2007.
- [23] R. Moser and G. Tardos. A constructive proof of the general Lovász Local Lemma. *Journal of the ACM*, 57(2):1–15, 2010.
- [24] W. Pegden. An extension of the Moser-Tardos algorithmic Local Lemma. *Arxiv 1102.2583*, 2011. To appear in the SIAM J. Discrete Mathematics.
- [25] B. Peis and A. Wiese. Universal packet routing with arbitrary bandwidths and transit times. In *IPCO*, pages 362–375, 2011.
- [26] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.
- [27] T. Rothvoss. A simpler proof for $O(\text{congestion} + \text{dilation})$ packet routing. In *Proc. Conference on Integer Programming and Combinatorial Optimization*, 2013. URL: <http://arxiv.org/pdf/1206.3718.pdf>.
- [28] C. Scheideler. Universal routing strategies for interconnection networks. In *Lecture Notes in Computer Science*, volume 1390. Springer, 1998.
- [29] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8:223–250, 1995.
- [30] M. Singh. *Iterative methods in combinatorial optimization*. PhD thesis, Tepper School of Business, Carnegie-Mellon University, 2008.
- [31] T. Szabó and G. Tardos. Extremal problems for transversals in graphs with bounded degree. *Combinatorica*, 26:333–351, 2006.
- [32] R. Yuster. Independent transversals in r -partite graphs. *Discrete Math.*, 176:255–261, 1997.

APPENDIX A. A TECHNICAL RESULT ON THE CHERNOFF SEPARATION FUNCTION

Proposition A.1. *Let $0 \leq \lambda \leq y$ and $\mu \geq 0$. Then we have*

$$Ch(\mu, (1 + \lambda)\mu) \leq e^{-\mu r \lambda^2}$$

where

$$r = \frac{-\ln\left(\frac{e^y}{(1+y)^{1+y}}\right)}{y^2}$$

Proof. We have that $\text{Ch}(\mu, (1 + \lambda)\mu) = \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^\mu = ((r(x)x^2)^{-\mu})$, where we define the function

$$r(x) = \frac{-\ln\left(\frac{e^x}{(1+x)^{1+x}}\right)}{x^2}$$

We have $r'(x) = \frac{2x-(2+x)\ln(1+x)}{x^3}$; we claim that $r'(x) \leq 0$ for $x \in [0, \infty)$. To show this suffices to show that $g(x) \leq 0$ where $g(x) = 2x - (2+x)\ln(1+x)$.

Now $g''(x) = \frac{-x}{(1+x)^2}$, which is clearly ≤ 0 for $x \geq 0$. So $g'(x) \leq g'(0) = 0$. So $g'(x) \leq 0$ for $x \geq 0$. So $g(x) \leq g(0) = 0$.

So, we have shown that $r'(x) \leq 0$. This implies that $r(\lambda) \geq r(y)$ so So

$$\text{Ch}(\mu, (1 + \lambda)\mu) = (r(\lambda)\lambda^2)^{-\mu} \leq (r(y)\lambda^2)^{-\mu} = e^{-\mu r(y)\lambda^2}$$

□